On the Complexity of Resource-Bounded Logics

N. Alechina[1], N. Bulling[2], S. Demri[3] and B. Logan[1]

[1] University of Nottingham [2] TU Delft [3] LSV, CNRS, ENS Cachan

Abstract. We revisit decidability results for resource-bounded logics and use decision problems on VASS in order to establish complexity characterisation of (decidable) model-checking problems. We show that the model-checking problem for RB±ATL is 2EXPTIME-complete by using recent results on alternating VASS (and in EXPTIME when the number of resources is bounded). Moreover, we establish that the model-checking problem for RBTL is EXPSPACE-complete and the problem is decidable and of the same complexity for RBTL*, proving a new decidability result as a by-product of the approach. When the number of resources is bounded, this problem is in PSPACE. We also establish that the model-checking problem for RB±ATL*, the extension of RB±ATL with arbitrary path formulae is decidable by reduction into parity games for single-sided VASS. Furthermore, we are able to synthesise values for resource parameters. Hence, the paper establishes formal correspondences between model-checking problems for resource-bounded logics and decision problems on alternating VASS, paving the way to more applications.

1 Introduction

Resource-bounded logics. Alternating-time temporal logics such as the logics ATL and ATL* [AHK02] extend the temporal logics CTL and CTL* respectively, by interpreting the formulae on concurrent game structures, a sophisticated extension of labelled transition systems, and by allowing modalities to quantify over strategies for a given coalition of agents. The logics ATL and ATL* are indeed well-established formalisms to reason about multi-agent transition systems and many variants have flourished over the years, see e.g. [LMO08,ALNR15]. The logic ATL significantly extends CTL by allowing strategy modalities while computational properties of the model-checking problem remains quite stable. For instance, in [AHK02], the labeling algorithm for model-checking CTL is extended to ATL leading to the P-completeness of the model-checking problem for ATL [AHK02, Theorem 5.2]. By contrast, the model-checking problem for ATL* is 2EXPTIME-complete [AHK02, Theorem 5.6] whereas the problem for CTL* is only PSPACE-complete, see e.g. [Eme90].

The logic ATL has many extensions, some of them have models containing transitions that produce or consume resources, depending on the actions involved to fire transitions, see e.g. the *resource-bounded logics* in [BF09,MNP11,ALNR14,ABLN15]. As shown in [ABLN15], having implicitly in the models counters (i.e. variables interpreted over the set of natural numbers) and the ability to quantify over strategies for a given coalition (i.e. for a set of agents) can easily lead to undecidability even though many logics have been introduced for which the model-checking problem is shown to

be decidable, possibly with a very high worst-case complexity upper bound. Reasoning about resources happens to be quite similar to the analysis of runs of vector addition systems with states (a.k.a. VASS) [KM69] and more specifically with games on VASS, see e.g. [BJK10], because of the very nature of strategy modalities. VASS, and more generally counter machines, are well-known infinite-state systems with many applications in formal verification, see e.g. [BBH+06].

Model-checking and games on VASS. In this work, we follow the approach that consists in using as much as possible results on VASS in order to analyse the model-checking problem for resource-bounded logics. But, as we recall below, model-checking problems on VASS based on temporal logics and games are not always decidable or at least quite difficult to solve but sharp results exist. Let us recapitulate a few facts below.

Temporal logics on VASS often lead to undecidable model-checking problems, see e.g. [Esp94,Esp98] and this is even more true with branching-time temporal logics such as CTL [Esp98] or when the atomic formulae can state properties about the counter values [HR89]. There are still exceptions for decidability. For instance, CTL model-checking on one-counter VASS is PSPACE-complete [Ser06,GL13] (see also [Ves15]), the control state repeated reachability problem for VASS is shown to be decidable in [Jan90] and this is generalised to full LTL (for which the atomic formulae are exactly control states) and more precisely the model-checking problem for LTL on VASS is EXPSPACE-complete [Hab97]. In [Jan90], a strict fragment of LTL restricted to the "infinitely often" temporal operator GF and atomic formulae stating properties on counter values is also shown decidable by reduction into the reachability problem for VASS.

As far as games for VASS are concerned, the situation is even less pleasant. Indeed, two-players games on VASS in which each player can freely update the counter values are undecidable [BJK10] even with simple winning conditions such as the reachability of a given control state. However, decidability of VASS games has been observed with asymmetric games in which at most one player can freely update the counter values [RSB05] and the winning conditions are simple. For instance, the game on asymmetric VASS (slight variants of single-sided VASS in [BHSS12,AMSS13] or alternating VASS in [CS14]) with reachability of a control state is shown to be 2EXPTIME-complete in [CS14] and decidable with parity conditions [AMSS13]. Non-termination problem for games on asymmetric games is also 2EXPTIME-complete (the upper bound is from [JLS15] and lower bound is from [CS14]).

Our motivations. Our main goal in this paper is to establish formal relationships between model-checking problems for resource-bounded logics and decision problems for VASS so that new decidability results can be established for logical problems or new complexity characterisations can be inherited from problems on counter machines. Of course, this should not come as a real surprise because resource values and counter values are similar objects and logics based on concurrent game structures have inherently games in the semantics. Moreover, earlier works have already explored the connections with counter machines, either to obtain undecidability results or to get complexity lower bounds, see e.g. [ABLN15]. Herein, we aim at obtaining optimal complexity upper bounds and new decidability results even for resource-bounded logics with enriched path formulae as those in CTL* [Eme90].

Our contributions. As mentioned earlier, our approach can be summarised as follows: to use results on decision problems for alternating VASS (a.k.a. single-sided VASS) in order to decide optimally model-checking problems on resource-bounded logics. So far, the reductions were rather in the other direction to establish undecidability (for instance, by reducing the halting problem for Minsky machines).

- The model-checking problem for RB±ATL is shown to be 2EXPTIME-complete. The restriction to a bounded number of resources is also shown in EXPTIME. The 2EXPTIME lower bound is obtained by reduction from the state reachability problem for AVASS [CS14] whereas the upper bound is by reduction into the state reachability and the termination problems for AVASS. Decision procedures for both problems are really needed here. So far, the best known result was decidability established in [ALNR14] by taking advantage of the well-quasi-ordering $(\mathbb{N}^r, \preceq)$.
- The above results are obtained by using formal relationships between strategies in resource-bounded concurrent game structures and proofs in alternating VASS and the fact that asymmetric VASS are only needed here is the key observation. These relationships are once more instrumental to show that the model-checking problem for RB±ATL* (a new logic naturally extending RB±ATL, as ATL* naturally extends ATL [AHK02]) is decidable by reduction into the parity game problem on single-sided VASS [AMSS13]. Note that the complexity characterisation of the parity game problem on single-sided VASS is still open as far as we know, see e.g. [AMSS13,CS14,JLS15]. More importantly, we show that resource parameters can be effectively computed in the parameterised version of RB±ATL* thanks to the fact that the Pareto frontier for any parity game on single-sided VASS is computable [AMSS13, Theorem 4]. As far as we know, this is the first time that resource values are synthetised in resource-bounded logics and this is done for the quite rich new logic RB±ATL*.
- The model-checking problem for RBTL [BF09] is proved EXPSPACE-complete. The restriction to a bounded number of resources is also shown in PSPACE. The model-checking problem for RBTL* is shown decidable (a new result) but also EXPSPACE-complete.

2 Logical Preliminaries

We write $\mathbb{N}$ [resp. $\mathbb{Z}$] for the set of natural numbers [resp. integers] and $[m, m']$ with $m, m' \in \mathbb{Z}$ to denote the set $\{j \in \mathbb{Z} : m \leq j \leq m'\}$. Given a dimension $r \geq 1$ and $a \in \mathbb{Z}$, we write $\boldsymbol{a} \in \mathbb{Z}^r$ to denote the vector with all values equal to a. For each $\boldsymbol{x} \in \mathbb{Z}^r$, we write $\boldsymbol{x}(1), \ldots, \boldsymbol{x}(r)$ for the entries of $\boldsymbol{x}$. For each $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{Z}^r$, $\boldsymbol{x} \preceq \boldsymbol{y} \stackrel{\text{def}}{\Leftrightarrow}$ for every $i \in [1, r]$, we have $\boldsymbol{x}(i) \leq \boldsymbol{y}(i)$. We also write $\boldsymbol{x} \prec \boldsymbol{y}$ when $\boldsymbol{x} \preceq \boldsymbol{y}$ and $\boldsymbol{x} \neq \boldsymbol{y}$.

2.1 The logic RB±ATL and its variants

Let PROP be a countably infinite set of atomic propositions. The models for the logics RB±ATL and RB±ATL* are the structures introduced in Definition 1 below. These are concurrent game structures for the logics ATL or ATL* (see e.g. [AHK02]) but enriched

with a cost function that allows to simulate how resources are produced or consumed. At some abstract level, a structure is equipped with r counters and the transitions update their values with increments or decrements.

Definition 1. *A* resource-bounded concurrent game structure $\mathfrak{M}$ *is a tuple*

$$\mathfrak{M} = (Agt, S, Act, r, \mathtt{act}, \mathtt{cost}, \delta, L)$$

such that:

- Agt *is a non-empty finite set of* agents *(by default* $Agt = [1, k]$ *for some* $k \geq 1$*).*
- S *is a set of states.*
- Act *is a non-empty set of actions with a distinguished action* `idle`*.*
- $r \geq 1$ *is the number of resources.*
- $\mathtt{act} : Agt \times S \to \mathcal{P}(Act)$ *is the* action manager function *such that for all* a *and* s*,* $\mathtt{act}(a, s)$ *is non-empty and furthermore we have* $\mathtt{idle} \in \mathtt{act}(a, s)$*.*
- $\mathtt{cost} : S \times Agt \times Act \to \mathbb{Z}^r$ *is the (partial) cost function so that* $\mathtt{cost}(s, a, \mathtt{a})$ *is defined exactly when* $\mathtt{a} \in \mathtt{act}(a, s)$*. Moreover,* $\mathtt{cost}(s, a, \mathtt{idle}) = \mathbf{0}$*.*
- $\delta : S \times (Agt \to Act) \to S$ *is the (partial) transition function such that* δ *is defined on* $(s, \mathfrak{f})$ *whenever for all agents* $a \in Agt$*, we have* $\mathfrak{f}(a) \in \mathtt{act}(a, s)$*.*
- $L : \mathrm{PROP} \to \mathcal{P}(S)$ *is a truth assignment (the definition can be adapted when finite subsets of* PROP *are involved).*

The map δ is also viewed as a deterministic transition relation with transitions of the form $s \xrightarrow{(\mathtt{a}_1, \ldots, \mathtt{a}_k)} s'$ where $\delta(s, \mathfrak{f}) = s'$ and for all $i \in [1, k] = Agt$, we have $\mathfrak{f}(i) = \mathtt{a}_i$. We say that $\mathfrak{M}$ is finite whenever S is a finite set and L is restricted to a finite subset of PROP. The presence of the idle action has been initially introduced in [ALNR14,ALNR15], where can be found also motivations for considering such a distinguished action. Given a *coalition* $A \subseteq Agt$ and a state s, a *joint action* by A is a map $\mathfrak{f} : A \to Act$ such that for all agents $a \in A$, we have $\mathfrak{f}(a) \in \mathtt{act}(s, a)$. The set of joint actions by A is denoted $D_A(s)$. Given a state s, the set of joint actions by Agt is simply denoted $D(s)$ (instead of $D_{Agt}(s)$) and the map δ is defined only for such joint actions. We write $\mathfrak{f} \sqsubseteq \mathfrak{g}$ whenever $\mathfrak{g}$ is a conservative extension of $\mathfrak{f}$ and the transition function δ takes as arguments joint actions by Agt.

Given a joint action $\mathfrak{f} \in D_A(s)$, we write $\mathtt{out}(s, \mathfrak{f})$ to denote the set below:

$$\mathtt{out}(s, \mathfrak{f}) \stackrel{\text{def}}{=} \{s' \in S \mid \text{there is } \mathfrak{g} \in D(s) \text{ such that } \mathfrak{f} \sqsubseteq \mathfrak{g} \text{ and } s' = \delta(s, \mathfrak{g})\}.$$

For instance, $\mathtt{out}(s, \mathfrak{f})$ is a singleton set when $\mathfrak{f} \in D(s)$ since δ is a map and not a relation. Given a joint action $\mathfrak{f} \in D_A(s)$ and a state s, the *cost* of any transition fired from s following $\mathfrak{f}$ (restricted to A by definition) is defined as follows:

$$\mathtt{cost}_A(s, \mathfrak{f}) \stackrel{\text{def}}{=} \sum_{a \in A} \mathtt{cost}(s, a, \mathfrak{f}(a)).$$

In a sense, the value $\mathtt{cost}_A(s, \mathfrak{f})$ does not depend on the costs related to the agents in $(Agt \setminus A)$, or equivalently, the cost related to the agents in $(Agt \setminus A)$ is reduced to zero.

A *computation* λ is a finite sequence or an ω-sequence of the form $s_0 \xrightarrow{\mathfrak{f}_0} s_1 \xrightarrow{\mathfrak{f}_1} s_2 \dots$ such that for all $i < |\lambda|$, we have $s_{i+1} \in \delta(s_i, \mathfrak{f}_i)$. Here, $|\lambda|$ denotes the length of λ, each s_i is a state and each $\mathfrak{f}_i$ belongs to $D(s_i)$. For instance $|s_0 \xrightarrow{\mathfrak{f}_0} s_1 \cdots \xrightarrow{\mathfrak{f}_{n-1}} s_n| = n + 1$ and $|s_0 \xrightarrow{\mathfrak{f}_0} s_1 \cdots \xrightarrow{\mathfrak{f}_{n-1}} \cdots| = \omega$ for any infinite computation.

So, in full generality, in a computation, a transition between two successive states are labelled by a joint action: this is not strictly needed for the forthcoming developments but this provides a more general notion that might be used in other contexts (for instance, if the winning condition of forthcoming strategies depends on the actions of all the agents and not only on those for the agents in A or on the visited states). A *strategy* F_A for the coalition A is a map from the set of finite computations to the set of joint actions by A such that

$$F_A(s_0 \xrightarrow{\mathfrak{f}_0} s_1 \cdots \xrightarrow{\mathfrak{f}_{n-1}} s_n) \in D_A(s_n).$$

A computation $\lambda = s_0 \xrightarrow{\mathfrak{f}_0} s_1 \xrightarrow{\mathfrak{f}_1} s_2 \cdots$ respects the strategy F_A iff for all $i < |\lambda|$, we have, $s_{i+1} \in \mathtt{out}(s_i, F_A(s_0 \xrightarrow{\mathfrak{f}_0} s_1 \dots \xrightarrow{\mathfrak{f}_{i-1}} s_i))$. A computation λ that respects F_A is *maximal* whenever it cannot be extended further while respecting the strategy. Note that maximal computations respecting F_A are infinite. The set of all maximal computations that respect the strategy F_A and that start at the state s is denoted by $\mathtt{out}(s, F_A)$. So far, no resource value has been involved in computations. Below, we shall quantify over maximal computations that respect a strategy and therefore for defining a strategy we can restrict ourselves to finite computations that respect it so far.

Given a bound $\boldsymbol{b} \in (\mathbb{N} \cup \{\omega\})^r$, a computation $\lambda = s_0 \xrightarrow{\mathfrak{f}_0} s_1 \xrightarrow{\mathfrak{f}_1} s_2 \dots$ in $\mathtt{out}(s, F_A)$ is $\boldsymbol{b}$-*consistent* iff for all $i < |\lambda|$, we have

$$\mathbf{0} \leq (\sum_{j=0}^{i-1} \mathtt{cost}_A(s_j, F_A(s_0 \xrightarrow{\mathfrak{f}_0} s_1 \dots \xrightarrow{\mathfrak{f}_{j-1}} s_j)) + \boldsymbol{b}).$$

Whenever $\boldsymbol{b}(i) = \omega$, this can be viewed as a means to disregard what happens on the ith resource (assuming that $n + \omega = \omega$ for any $n \in \mathbb{Z}$). Indeed, $\boldsymbol{b}(i) = \omega$ amounts to guarantee from the beginning of the computation that there is an infinite supply of resources on the ith component. Note also that the above condition is slightly different from the one in [ALNR15] but strictly equivalent. We have decided to adopt that notation in order to show more easily the relationships with VASS decision problems. So, from a computation $\lambda = s_0 \xrightarrow{\mathfrak{f}_0} s_1 \xrightarrow{\mathfrak{f}_1} s_2 \dots$, there is an underlying sequence $\boldsymbol{v}_0, \boldsymbol{v}_1, \dots$ of resource values so that $\boldsymbol{v}_0 \stackrel{\text{def}}{=} \boldsymbol{b}$ and for all $i < |\lambda|$, we have $\boldsymbol{v}_{i+1} \stackrel{\text{def}}{=} \boldsymbol{v}_i + \mathtt{cost}_A(s_i, F_A(s_0 \xrightarrow{\mathfrak{f}_0} s_1 \dots \xrightarrow{\mathfrak{f}_{i-1}} s_i))$. Since the values of the sequence only depend on the agents in A, this hypothesis is often called the *proponent restriction condition*.

The set of all the $\boldsymbol{b}$-consistent (infinite) computations is denoted by $\mathtt{out}(s, F_A, \boldsymbol{b})$. A $\boldsymbol{b}$-*strategy* F_A *with respect to* s is a strategy such that $\mathtt{out}(s, F_A) = \mathtt{out}(s, F_A, \boldsymbol{b})$. This definition also slightly differs from the one in [ALNR15] that is not relative to a given state and therefore in [ALNR15] the equality should hold for all the states.

So far, we have provided the main definitions about resource-bounded concurrent game structures and strategies. Let us present now the logic RB±ATL (the logic RB±ATL* will be presented in Section 5.2). Given a set of agents $Agt = \{a_1, \dots, a_k\}$ and $r \geq 1$,

we write RB±ATL(Agt, r) to denote the resource-bounded logic with k agents and r resources whose models are resource-bounded concurrent game structures with the same parameters. Formulae of RB±ATL(Agt, r) are defined according to the grammar below:

$$\phi ::= p \mid \neg\phi \mid \phi \wedge \phi \mid \langle\langle A^{\boldsymbol{b}}\rangle\rangle \bigcirc \phi \mid \langle\langle A^{\boldsymbol{b}}\rangle\rangle \Box\phi \mid \langle\langle A^{\boldsymbol{b}}\rangle\rangle \phi \boldsymbol{\mathcal{U}} \phi,$$

where $p \in \text{PROP}$, $A \subseteq Agt$ and $\boldsymbol{b} \in (\mathbb{N} \cup \{\omega\})^r$. The size of a formula is computed from a DAG representation and the integers are encoded in binary. Note that forthcoming hardness results do not use the conciseness of the DAG representation (with respect to the tree representation). The satisfaction relation $\models$ is defined inductively as follows assuming that $\mathfrak{M}$ is an RB±ATL(Agt, r) model (we omit the obvious cases for the Boolean connectives):

$$\begin{array}{lcl}
\mathfrak{M}, s \models p & \stackrel{\text{def}}{\Leftrightarrow} & s \in L(p) \\
\mathfrak{M}, s \models \langle\langle A^{\boldsymbol{b}}\rangle\rangle \bigcirc \phi & \stackrel{\text{def}}{\Leftrightarrow} & \text{there is a } \boldsymbol{b}\text{-strategy } F_A \text{ w.r.t. } s \text{ such that} \\
 & & \text{for all } s_0 \xrightarrow{\mathfrak{f}_0} s_1 \ldots \in \mathtt{out}(s, F_A), \text{ we have } \mathfrak{M}, s_1 \models \phi \\
\mathfrak{M}, s \models \langle\langle A^{\boldsymbol{b}}\rangle\rangle \Box\phi & \stackrel{\text{def}}{\Leftrightarrow} & \text{there is a } \boldsymbol{b}\text{-strategy } F_A \text{ w.r.t. } s \text{ such that} \\
 & & \text{for all } \lambda = s_0 \xrightarrow{\mathfrak{f}_0} s_1 \ldots \in \mathtt{out}(s, F_A), \text{ for all } i < |\lambda|, \\
 & & \text{we have } \mathfrak{M}, s_i \models \phi \\
\mathfrak{M}, s \models \langle\langle A^{\boldsymbol{b}}\rangle\rangle \phi_1 \boldsymbol{\mathcal{U}} \phi_2 & \stackrel{\text{def}}{\Leftrightarrow} & \text{there is a } \boldsymbol{b}\text{-strategy } F_A \text{ w.r.t. } s \text{ such that for all} \\
 & & \lambda = s_0 \xrightarrow{\mathfrak{f}_0} s_1 \ldots \in \mathtt{out}(s, F_A), \text{ there is some } i < |\lambda| \text{ such that} \\
 & & \mathfrak{M}, s_i \models \phi_2 \text{ and for all } j \in [0, i-1], \text{ we have } \mathfrak{M}, s_j \models \phi_1.
\end{array}$$

It is worth noting that since all the maximal computations are infinite, the index i involved for clauses related to $\langle\langle A^{\boldsymbol{b}}\rangle\rangle\Box$ or $\langle\langle A^{\boldsymbol{b}}\rangle\rangle \cdot \boldsymbol{\mathcal{U}} \cdot$ can take any value in $\mathbb{N}$. The "temporal operators" $\bigcirc$, $\Box$ and $\boldsymbol{\mathcal{U}}$ have their standard meaning from linear-time temporal logic LTL. The presence of the idle action allows to extend a strategy as soon as a given condition needs to be satisfied along the computations. For instance, $\mathfrak{M}, s \models \langle\langle A^{\boldsymbol{b}}\rangle\rangle \bigcirc \phi$ is equivalent to the existence of $\mathfrak{f} \in D_A(s)$ such that for all $\mathfrak{g} \sqsupseteq \mathfrak{f}$, we have $\mathfrak{M}, s' \models \phi$ with $\delta(s, \mathfrak{g}) = s'$ and $\mathbf{0} \preceq \boldsymbol{b} + \mathtt{cost}_A(s, \mathfrak{f})$.

Observe also that a strategy modality $\langle\langle A^{\boldsymbol{b}}\rangle\rangle$ reduces the impact of the function `cost` in two ways. If the ith component of $\boldsymbol{b}$ is equal to ω, then there are no constraints on the ith resource along the computation. Moreover, the restriction of `cost` to opponent agents in $(Agt \setminus A)$ is reduced to $\mathbf{0}$ (so without any impact on consistency). Besides, it is worth noting that $\langle\langle A^{\boldsymbol{b}}\rangle\rangle \phi \boldsymbol{\mathcal{U}} \psi \Rightarrow \langle\langle A^{\boldsymbol{b}'}\rangle\rangle \phi \boldsymbol{\mathcal{U}} \psi$ is valid whenever $\boldsymbol{b} \preceq \boldsymbol{b'}$ and therefore whenever $\mathfrak{M}, s \models \langle\langle A^{\boldsymbol{b}}\rangle\rangle \phi \boldsymbol{\mathcal{U}} \psi$ there is a finite set of minimal elements $\boldsymbol{m} \in (\mathbb{N} \cup \{\omega\})^r$ (with respect to $\preceq$) such that $\mathfrak{M}, s \models \langle\langle A^{\boldsymbol{m}}\rangle\rangle \phi \boldsymbol{\mathcal{U}} \psi$ (by Dickson's Lemma [Dic13] every upward closed set of $(\mathbb{N} \cup \{\omega\})^r$ admits a finite basis of minimal elements), see also the notion of *Pareto frontier* in Section 5.2.

Obviously, RB±ATL(Agt, r) is a quantitative variant of ATL [AHK02] in which resource values are computed along the computations and the logic RB±ATL is introduced in [ALNR14,ALNR15]. The *model-checking problem for RB±ATL* is defined as follows:

Input: $k, r \geq 1$ (in unary), a formula ϕ in RB±ATL($[1, k], r$), a finite RB±ATL($[1, k], r$) model $\mathfrak{M}$ and a state s,

Question: $\mathfrak{M}, s \models \phi$?

The encoding of the values in k and r in unary is unessential here since the size of $\mathfrak{M}$ with an explicit representation of all the transitions is greater than $k + r$.

Proposition 1. *[ALNR14, Theorem 1] The model-checking problem for RB±ATL is decidable.*

An important part of the paper is dedicated to characterising the computational complexity of the model-checking problem for RB±ATL.

3 Problems on Vector Addition Systems with States (VASS)

In this section, we recall known complexity/decidability results about model-checking and games on VASS and we state the necessary complexity characterisations that will be used in the sequel. Part of the paper is dedicated to show that using optimal decision procedures for such problems as black boxes lead to optimal decision problems for model-checking problems on resource-bounded logics.

3.1 Alternating VASS

A binary tree $\mathfrak{T}$, which may contain nodes with one child, is a non-empty subset of $\{1,2\}^*$ such that, for all $\mathfrak{n} \in \{1,2\}^*$ and $i \in \{1,2\}$, $\mathfrak{n} \cdot i \in \mathfrak{T}$ implies $\mathfrak{n} \in \mathfrak{T}$ and, $\mathfrak{n} \cdot 2 \in \mathfrak{T}$ implies $\mathfrak{n} \cdot 1 \in \mathfrak{T}$. The nodes of $\mathfrak{T}$ are its *elements*. The root of $\mathfrak{T}$ is ε, the empty word. All notions such as parent, first child, second child, subtree and leaf, have their standard meanings. The height of $\mathfrak{T}$ is the length, i.e. the number of nodes, of the longest simple path from the root to a leaf. An *alternating VASS* (AVASS) [CS14] is a tuple $\mathcal{A} = (Q, r, R_1, R_2)$ such that

- Q is a finite set of *locations* (a.k.a. *control states*) and $r \geq 0$ is the number of resource values,
- R_1 is a finite subset of $Q \times \mathbb{Z}^r \times Q$ (*unary rules*),
- R_2 is a (finite) subset of Q^3 (*fork rules*).

A *derivation skeleton* of $\mathcal{A}$ is a labelling $\mathcal{D} : \mathfrak{T} \to (R_1 \cup R_2 \cup \{\bot\})$ such that:

- $\mathfrak{T}$ is a binary tree,
- if $\mathfrak{n}$ has one child in $\mathfrak{T}$, then $\mathcal{D}(\mathfrak{n}) \in R_1$,
- if $\mathfrak{n}$ has two children in $\mathfrak{T}$, then $\mathcal{D}(\mathfrak{n}) \in R_2$,
- if $\mathfrak{n}$ is a leaf in $\mathfrak{T}$, then $\mathcal{D}(\mathfrak{n}) = \bot$.

A *derivation* of $\mathcal{A}$ based on $\mathcal{D}$ is a labelling $\hat{\mathcal{D}} : \mathfrak{T} \to Q \times \mathbb{Z}^r$ such that:

- if $\mathfrak{n}$ has one child $\mathfrak{n}'$ in $\mathfrak{T}$, $\mathcal{D}(\mathfrak{n}) = (q, \boldsymbol{u}, q')$ and $\hat{\mathcal{D}}(\mathfrak{n}) = (q, \boldsymbol{v})$, then $\hat{\mathcal{D}}(\mathfrak{n}') = (q', \boldsymbol{v} + \boldsymbol{u})$.
- if $\mathfrak{n}$ has two children $\mathfrak{n}'$ and $\mathfrak{n}''$ in $\mathfrak{T}$, $\mathcal{D}(\mathfrak{n}) = (q, q_1, q_2)$ and $\hat{\mathcal{D}}(\mathfrak{n}) = (q, \boldsymbol{v})$, then $\hat{\mathcal{D}}(\mathfrak{n}') = (q_1, \boldsymbol{v})$ and $\hat{\mathcal{D}}(\mathfrak{n}'') = (q_2, \boldsymbol{v})$.

So, fork rules do no update the resources and whence, there is an asymmetry between unary rules and fork rules. This will be a much useful feature when dealing with the proponent restriction condition in RB±ATL. Unlike branching VASS (see e.g. [VGL05,DJLL13]), the fork rules have no effect on the counter values.

A derivation $\hat{\mathcal{D}}$ is *admissible* whenever $\hat{\mathcal{D}} : \mathfrak{T} \to Q \times \mathbb{N}^r$, i.e. only natural numbers occur in it. An admissible derivation is also called a *proof*. Earlier, we have introduced the primitive notion of computations and then one can restrict oneself to $\boldsymbol{b}$-consistent ones. Similarly, there is the primitive notion of derivations and then one can restrict oneself to proofs (a kind of "$\mathbf{0}$-consistency").

Before presenting the decision problems on AVASS, we would like to state a simple property that will be used in the sequel.

Lemma 1. *Given a derivation skeleton $\mathcal{D} : \mathfrak{T} \to (R_1 \cup R_2 \cup \{\bot\})$ and $(q, \boldsymbol{b}) \in Q \times \mathbb{Z}^r$, there is a unique derivation $\hat{\mathcal{D}}$ of $\mathcal{A}$ based on $\mathcal{D}$ such that $\hat{\mathcal{D}}(\varepsilon) = (q, \boldsymbol{b})$.*

Indeed, once the rules are provided thanks to $\mathcal{D}$, the root value $(q, \boldsymbol{b})$ determines all the values of the derivation since the way $\mathcal{D}(\mathfrak{n})$ is defined remains essentially deterministic. The *state reachability problem* for AVASS is defined as follows:

Input: An alternating VASS $\mathcal{A}$ and control states q_0 and q_f.
Question: Is there is a finite proof of AVASS whose root is equal to $(q_0, \mathbf{0})$ and each leaf belongs to $\{q_f\} \times \mathbb{N}^r$?

When $\mathcal{A}$ has no fork rules, $\mathcal{A}$ is essentially a VASS [KM69] and the above problem is an instance of the coverability problem known to be EXPSPACE-complete [Lip76,Rac78] (see also [BS11,Dem13]). The *non-termination problem* for AVASS is defined as follows:

Input: An alternating VASS $\mathcal{A}$ and a control state q_0.
Question: Is there is a proof whose root is equal to $(q_0, \mathbf{0})$ and all the maximal branches are infinite?

Proposition 2. *[CS14,JLS15] The state reachability and non-termination problems for AVASS are* 2EXPTIME*-complete.*

Decidability of these problems were first established in [RSB05] by using monotonicity of the games. The 2EXPTIME upper bound is preserved if we assume that the root is labelled by $(q_0, \boldsymbol{b})$ with $\boldsymbol{b} \in \mathbb{N}^r$ encoded with a binary representation (see Lemma 7).

In the sequel, we shall also admit fork rules of any arity $\alpha \geq 1$ and therefore in such slightly extended AVASS, the set of fork rules R_2 is a finite subset of $\bigcup_{\beta \geq 2} Q^\beta$. The notions of derivation skeleton, derivation and proof are also updated so that general trees $\mathfrak{T} \subseteq (\mathbb{N} \setminus \{0\})^*$ are involved. The set of finite words $\mathfrak{T} \subseteq (\mathbb{N} \setminus \{0\})^*$ is a (not necessarily binary) *tree* iff for all $\mathfrak{n} \in (\mathbb{N} \setminus \{0\})^*$ and $i \in (\mathbb{N} \setminus \{0\})$, $\mathfrak{n} \cdot i \in \mathfrak{T}$ implies $\mathfrak{n} \in \mathfrak{T}$ and, $\mathfrak{n} \cdot i \in \mathfrak{T}$ and $i > 1$ imply $\mathfrak{n} \cdot (i-1) \in \mathfrak{T}$. Such AVASS correspond to so-called *single-sided VASS* [BHSS12,AMSS13].

3.2 Model-checking problems

In the sequel, a VASS is understood as an alternating VASS without any fork rule and therefore we write it $\mathcal{V} = (Q, r, R)$ where R is a finite set of unary rules. Given a VASS $\mathcal{V}$, its transition system $\mathfrak{TS}(\mathcal{V}) \stackrel{\text{def}}{=} (\mathfrak{W}, \rightarrow, L)$ is such that:

- $\mathfrak{W} \stackrel{\text{def}}{=} Q \times \mathbb{N}^r$.
- L is a truth assigment with elements of Q also understood as propositional variables and $L(q) \stackrel{\text{def}}{=} \{q\} \times \mathbb{N}^r$.
- $\rightarrow$ is a binary relation on $\mathfrak{W}$ such that $(q, \boldsymbol{v}) \rightarrow (q', \boldsymbol{v}')$ iff there is a unary rule $(q, \boldsymbol{u}, q')$ in R such that $\boldsymbol{v}' = \boldsymbol{v} + \boldsymbol{u}$ where '+' is the component-wise addition on $\mathbb{N}^r$.

As usual, we also write $\stackrel{*}{\rightarrow}$ to denote the reflexive and transitive closure of $\rightarrow$. Since $\mathfrak{TS}(\mathcal{V})$ is a Kripke-style structure, it is possible to interpret on it modal or temporal formulae whose atomic formulae refer to control states. For instance, this includes formulae from the temporal logics LTL or CTL, to quote a few examples that will be considered in this document (see also [DDMM12]). Since alternating-time temporal logics such as ATL or ATL* extend strictly CTL or CTL* respectively, complexity hardness results for temporal logics can be lifted to such logics. We shall follow that path and we recall below the results that will be useful in the sequel.

Proposition 3. *The model-checking problem for LTL on VASS is* EXPSPACE*-complete (the atomic formulae are control states) and it is* PSPACE*-complete for a fixed number of resources [Hab97].*

EXPSPACE-hardness already holds for the state reachability problem for VASS [Lip76], which is a subproblem of the model-checking problem for VASS by considering the LTL formula $\Diamond q_f$.

3.3 The logic RBTL* and its variants

The models of the logic RBTL* are structures of the form (Q, r, R, L) where (Q, r, R) is a VASS and L is a truth assignment built on elements of Q understood as propositional variables, so that $L(q) = \{q\}$ (see e.g [BF09, Section 3]). In order to fit the usual terminology, below, an infinite proof in (Q, r, R) is called a *path* or *run* and it can be represented by $\lambda = (q_0, \boldsymbol{v}_0) \rightarrow (q_1, \boldsymbol{v}_1) \ldots$. We write $\lambda[+i, +\infty)$ to denote the run starting from $(q_i, \boldsymbol{v}_i)$ taken from λ as a suffix and $\lambda(i)$ to denote the configuration $(q_i, \boldsymbol{v}_i)$.

The *state formulae* ϕ and the *path formulae* Φ of RBTL* are defined by mutual recursion with the grammar (relatively to a set of locations Q and a number of resources r, which is fine since we are only interested in model-checking)

$$\begin{array}{rcl} \phi & ::= & q \mid \neg\phi \mid (\phi \wedge \phi) \mid \langle \boldsymbol{b} \rangle \Phi \\ \Phi & ::= & \phi \mid \neg\Phi \mid (\Phi \wedge \Phi) \mid \bigcirc \Phi \mid (\Phi \boldsymbol{\mathcal{U}} \Phi) \mid \Box\Phi \end{array}$$

where $q \in Q$. Syntactically every state formula is also a path formula according to this grammar, and this reflects the fact that a path uniquely identifies a control state in which

a formula is interpreted: its starting control state. We present the semantics for RBTL* in a style where the distinction between state formulae and path formulae is explicit. The two satisfaction relations $\models_{\mathfrak{s}}$ and $\models_{\mathfrak{p}}$ are defined as follows (standard clauses for the Boolean connectives are omitted).

$\mathfrak{M}, q \models_{\mathfrak{s}} q'$ iff $q' = q$
$\mathfrak{M}, q \models_{\mathfrak{s}} \langle \boldsymbol{b} \rangle \Phi$ iff there is an infinite run λ starting at $(q, \boldsymbol{b})$ such that $\mathfrak{M}, \lambda \models_{\mathfrak{p}} \Phi$
$\mathfrak{M}, \lambda \models_{\mathfrak{p}} \phi$ iff $\mathfrak{M}, \lambda(0) \models_{\mathfrak{s}} \phi$ for location formulae ϕ
$\mathfrak{M}, \lambda \models_{\mathfrak{p}} \bigcirc \Phi$ iff $\mathfrak{M}, \lambda[1, +\infty) \models_{\mathfrak{p}} \Phi$
$\mathfrak{M}, \lambda \models_{\mathfrak{p}} \Phi \mathcal{U} \Psi$ iff there is $i \geq 0$ such that $\mathfrak{M}, \lambda[i, +\infty) \models_{\mathfrak{p}} \Psi$ and
for every $j \in [0, i-1]$, we have $\mathfrak{M}, \lambda[j, +\infty) \models_{\mathfrak{p}} \Phi$.

The model-checking problem for RBTL* is defined as follows:

Input: a model $\mathfrak{M}$, a control state q and a state formula ϕ,
Question: $\mathfrak{M}, q \models_{\mathfrak{s}} \phi$?

As CTL is a syntactic fragment of CTL*, RBTL is defined as a syntactic fragment of RBTL* where each occurrence of a path modality $\langle \boldsymbol{b} \rangle$ is immediatly followed by a path formula whose outermost connective is among $\mathcal{U}$, $\bigcirc$ or $\Box$. Observe that the model-checking problem for RBTL is already EXPSPACE-hard since the state reachability problem for VASS can be reduced to a question of the form $\mathfrak{M}, q_0 \models \langle \mathbf{0} \rangle\, q_f$. Section 5.1 is dedicated to the computational complexity of the respective model-checking problems for RBTL* and RBTL.

4 On the Complexity of RB±ATL

4.1 Structural analysis about strategies and proofs

In this section, we establish formal relationships between strategies in resource-bounded concurrent game structures and proofs in associated alternating VASS. Conceptually, the technical developments are not extremely difficult but this will allow us to derive results that are helpful to solve the model-checking problem for RB±ATL thanks to decision procedures on AVASS. Actually, this is also useful to go beyond what is known today (see e.g. Section 5.2 and Section 5.3).

Let $\mathfrak{M}$ be a finite resource-bounded concurrent game structure, $A \subseteq Agt$ be a coalition, F_A be a strategy and s be a state. We construct an alternating VASS $\mathcal{A}_{\mathfrak{M},A,s}$ such that the set of computations respecting F_A and starting from s will correspond precisely to a derivation skeleton whose root is labelled by a unary rule with first state s. Moreover, if F_A is $\boldsymbol{b}$-consistent w.r.t. s, then the derivation skeleton can be turned into a proof whose root is labelled by $(s, \boldsymbol{b})$. This implies that fork rules can admit any arity greater than one and a component can admit the value ω, which is a value that remains constant then (so again we assume that $n + \omega = n$ for all $n \in \mathbb{Z}$).

Given $\mathfrak{M} = (Agt, S, Act, r, \mathtt{act}, \mathtt{cost}, \delta, L)$, the AVASS $\mathcal{A}_{\mathfrak{M},A,s} \stackrel{\text{def}}{=} (Q, r, R_1, R_2)$ is built as follows:

$$Q \stackrel{\text{def}}{=} \{s\} \cup \{(s', \mathfrak{f}) \mid s' \in S,\ \mathfrak{f} \in D_A(s')\} \cup \{(\mathfrak{g}, s') \mid s', s'' \in S,\ \mathfrak{g} \in D(s''),\ \delta(s'', \mathfrak{g}) = s'\}.$$

– The set of unary rules R_1 contains the following elements:
  - For all $\mathfrak{f} \in D_A(s)$, $(s, \mathtt{cost}_A(s, \mathfrak{f}), (s, \mathfrak{f}))$.
  - For all $(\mathfrak{g}, s') \in Q$, for all $\mathfrak{f} \in D_A(s')$, $((\mathfrak{g}, s'), \mathtt{cost}_A(s', \mathfrak{f}), (s', \mathfrak{f}))$.
– The set of fork rules R_2 contains the following elements.
  - For all $(s', \mathfrak{f}) \in Q$, let $\{(\mathfrak{g}_1, s_1), \ldots, (\mathfrak{g}_\alpha, s_\alpha)\} = \{(\mathfrak{g}, s'') \in S \mid s'' \in \delta(s', \mathfrak{g}),\ \mathfrak{g} \in D(s'),\ \mathfrak{f} \sqsubseteq \mathfrak{g}\}$. The set is non-empty thanks to constraints on the idle action (for instance). We add the α-ary fork rule $((s', \mathfrak{f}), (\mathfrak{g}_1, s_1), \ldots, (\mathfrak{g}_\alpha, s_\alpha))$. In order to define unambiguously that rule, we assume an arbitrary linear ordering on the set Q.

It is worth noting that:

– s has a special status in Q simply because any proof whose root configuration contains s has no predecessor configuration.
– Any derivation skeleton from $\mathcal{A}_{\mathfrak{M},A,s}$ has to alternate the rules in R_1 and the rules in R_2, by construction. This property will be used to slightly simplify developments below.
– For every $(s', \mathfrak{f})$ in Q, there is a unique fork rule starting by $(s', \mathfrak{f})$.
– The construction applies also in the degenerate cases, i.e. when $A = Agt$ or when $A = \emptyset$ (assuming that $\mathtt{cost}(s', \mathfrak{f}) = \mathbf{0}$ for the unique $\mathfrak{f} \in D_\emptyset(s')$).

In Figure 1, we illustrate how transitions from the state s are turned into unary rules and fork rules (with $Agt = \{1, 2\}$ and $A = \{1\}$).

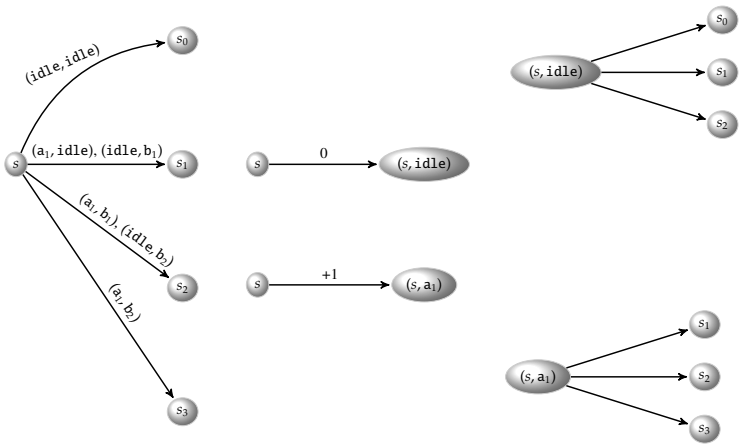

Fig. 1. Transitions and its associated unary and fork rules (+1 is arbitrary)

So, given an infinite computation $\lambda = s_0 \xrightarrow{\mathfrak{g}_1} s_1 \xrightarrow{\mathfrak{g}_2} s_2 \ldots$ respecting F_A and starting at s, we can associate the following infinite sequence (herein called an *extended computation*)

$$\mathtt{ext}(\lambda, F_A) \stackrel{\text{def}}{=} s_0 \xrightarrow{\boldsymbol{u}_0} (s_0, \mathfrak{f}_0) \rightarrow (\mathfrak{g}_1, s_1) \xrightarrow{\boldsymbol{u}_1} (s_1, \mathfrak{f}_1) \rightarrow (\mathfrak{g}_2, s_2) \xrightarrow{\boldsymbol{u}_2} (s_2, \mathfrak{f}_2) \rightarrow (\mathfrak{g}_3, s_3) \cdots$$

where $s_0 = s$ and for all $n \geq 0$, $F_A(s_0 \xrightarrow{\mathfrak{g}_1} s_1 \ldots \xrightarrow{\mathfrak{g}_n} s_n) = \mathfrak{f}_n$ and $\mathtt{cost}_A(s_n, \mathfrak{f}_n) = \boldsymbol{u}_n$. All the computations in $\mathtt{out}(s, F_A)$ can be organised as an infinite tree that corresponds to a derivation skeleton for $\mathcal{A}_{\mathfrak{M},A,s}$.

So below we define an infinite tree $\mathfrak{T}_{F_A}$, a labelling function $\mathfrak{L} : \mathfrak{T}_{F_A} \to S$ and a partial map $\mathfrak{R} : \mathfrak{T}_{F_A} \times \mathfrak{T}_{F_A} \to (\bigcup_{s' \in S} D(s'))$.

- $\mathfrak{T}_{F_A}(\varepsilon) \stackrel{\text{def}}{=} s_0$.
- For all finite words $\mathfrak{w} = k_1 \cdots k_\beta$ in $\mathfrak{T}_{F_A}$ such that $\mathfrak{L}(\mathfrak{w})$ is already defined, we add to $\mathfrak{T}_{F_A}$ the values $k_1 \cdots k_\beta \cdot 1, \ldots, k_1 \cdots k_\beta \cdot \alpha$ such that
  - $F_A(\mathfrak{L}(\varepsilon) \xrightarrow{\mathfrak{R}(\varepsilon,k_1)} \mathfrak{L}(k_1) \xrightarrow{\mathfrak{R}(k_1,k_1k_2)} \mathfrak{L}(k_1k_2) \cdots \xrightarrow{\mathfrak{R}(k_1\cdots k_{\beta-1},k_1\cdots k_\beta)} \mathfrak{L}(\mathfrak{w})) = \mathfrak{f}$
  - $\{(\mathfrak{g}_1, s_1), \ldots, (\mathfrak{g}_\alpha, s_\alpha)\} = \{(\mathfrak{g}, s'') \in S \mid s'' \in \delta(s', \mathfrak{g}),\ \mathfrak{g} \in D(s'),\ \mathfrak{f} \sqsubseteq \mathfrak{g}\}$.
  - For all $j \in [1, \alpha]$, we have $\mathfrak{L}(k_1 \cdots k_\beta \cdot j) \stackrel{\text{def}}{=} s_j$ and $\mathfrak{R}(\mathfrak{w}, \mathfrak{w} \cdot j) \stackrel{\text{def}}{=} \mathfrak{g}_j$.

The tree $\mathfrak{T}_{F_A}$ is defined by saturation of the above rules, and the maps $\mathfrak{L}$ and $\mathfrak{R}$ are defined accordingly. The structure $(\mathfrak{T}_{F_A}, \mathfrak{R}, \mathfrak{L})$ is indeed a labelled transition system with a tree-like structure encoding all the infinite computations respecting the strategy F_A. A maximal branch $\mathfrak{w}$ of $(\mathfrak{T}_{F_A}, \mathfrak{R}, \mathfrak{L})$ is understood as an element of $(\mathbb{N} \setminus \{0\})^\omega$ such that any (strict) finite prefix of $\mathfrak{w}$ belongs to $\mathfrak{T}_{F_A}$. The label of $\mathfrak{w}$, written $\mathtt{lab}(\mathfrak{w})$, is defined as follows where $\mathfrak{w} = k_1k_2k_3 \cdots$:

$$\mathtt{lab}(\mathfrak{w}) \stackrel{\text{def}}{=} \mathfrak{L}(\varepsilon) \xrightarrow{\mathfrak{R}(\varepsilon,k_1)} \mathfrak{L}(k_1) \xrightarrow{\mathfrak{R}(k_1,k_1k_2)} \mathfrak{L}(k_1k_2) \cdots \xrightarrow{\mathfrak{R}(k_1\cdots k_{\beta-1},k_1\cdots k_\beta)} \mathfrak{L}(k_1 \cdots k_\beta) \cdots$$

By construction, $\mathtt{lab}(\mathfrak{w})$ is a maximal computation.

Lemma 2.

(I) *For every maximal computation λ starting at s and respecting F_A, there is maximal branch $\mathfrak{w}$ in $(\mathfrak{T}_{F_A}, \mathfrak{R}, \mathfrak{L})$ such that $\lambda = \mathtt{lab}(\mathfrak{w})$.*

(II) *For every maximal branch $\mathfrak{w}$ in $(\mathfrak{T}_{F_A}, \mathfrak{R}, \mathfrak{L})$, there is a maximal computation λ starting at s and respecting F_A such that $\mathtt{lab}(\mathfrak{w}) = \lambda$.*

The proof is by an easy verification and it only reflects that $(\mathfrak{T}_{F_A}, \mathfrak{R}, \mathfrak{L})$ organizes all the computations from s that respects the strategy F_A.

Let us build the derivation skeleton $\mathcal{D} : \mathfrak{T}_{F_A} \to (R_1 \cup R_2)$ as follows where all the maximal branches of $\mathfrak{T}_{F_A}$ are infinite (so no need to include $\bot$ in the range of $\mathcal{D}$).

- $\mathcal{D}(\varepsilon) = (s_0, \mathtt{cost}_A(s_0, F_A(s_0)), (s_0, F_A(s_0)))$.
- $\mathcal{D}(1) = ((s_0, F_A(s_0)), (\mathfrak{g}_1, s_1), \ldots, (\mathfrak{g}_\alpha, s_\alpha))$ where $1, \ldots, \alpha \in \mathfrak{T}_{F_A}$ (but $\alpha + 1 \notin \mathfrak{T}_{F_A}$), for all $j \in [1, \alpha]$, $\mathfrak{T}_{F_A}(j) = s_j$ and $\mathfrak{R}(\varepsilon, j) = \mathfrak{g}_j$. By construction of $\mathcal{A}_{\mathfrak{M},A,s}$, $\mathcal{D}(1)$ is the unique fork rule starting by $(s_0, F_A(s_0))$.
- Let $\mathfrak{n} = 1k_11 \cdots k_\beta 1$ with $k_1, \ldots, k_\beta \geq 1$ and such that

  $$\mathcal{D}(1k_11 \cdots k_\beta) = ((\mathfrak{g}, s'), \mathtt{cost}_A(s', \mathfrak{f}), (s', \mathfrak{f})) \in R_1.$$

  Then, $\mathcal{D}(\mathfrak{n}) = ((s', \mathfrak{f}), (\mathfrak{g}_1, s_1), \ldots, (\mathfrak{g}_\alpha, s_\alpha))$ where $k_1 \cdots k_\beta 1, \ldots, k_1 \cdots k_\beta \alpha \in \mathfrak{T}_{F_A}$ (but $k_1 \cdots k_\beta(\alpha + 1) \notin \mathfrak{T}_{F_A}$), for all $j \in [1, \alpha]$,
  - $\mathfrak{T}_{F_A}(k_1 \cdots k_\beta \cdot j) = s_j$ and,
  - $\mathfrak{R}(k_1 \cdots k_\beta, k_1 \cdots k_\beta j) = \mathfrak{g}_j$.

By construction of $\mathcal{A}_{\mathfrak{M},A,s}$, $\mathcal{D}(\mathfrak{n})$ is the unique fork rule starting by $(s', \mathfrak{f})$.

– Let $\mathfrak{n} = 1k_1 1 \cdots k_\beta$. By construction we can assume that we already have that $\mathcal{D}(1k_1 1 \cdots k_{\beta-1}1) = ((s', \mathfrak{f}'), (\mathfrak{g}_1, s_1), \ldots, (\mathfrak{g}_\alpha, s_\alpha))$. Let $\mathfrak{g}'$ be equal to $\mathfrak{R}(k_1 \cdots k_\beta, k_1 \cdots k_\beta \cdot 1)$ and $\mathfrak{f}$ be the restriction of $\mathfrak{g}'$ to A (so $\mathfrak{f} \sqsubseteq \mathfrak{g}'$). Then,

$$\mathcal{D}(\mathfrak{n}) = ((\mathfrak{g}_{k_\beta}, s_{k_\beta}), \mathtt{cost}_A(s_{k_\beta}, \mathfrak{f}), (s_{k_\beta}, \mathfrak{f})).$$

Note that $\mathcal{D}(\mathfrak{n})$ is indeed a valid unary rule.

Given an infinite branch $\mathfrak{w}$ of $\mathcal{D}$, say $\mathfrak{w} = 1k_1 1k_2 1k_3 \cdots \in \mathbb{N}^\omega$, we define the extended computation $\mathtt{ext}(\mathfrak{w}, F_A)$ as follows. Suppose that the label of such a branch is characterised by the values below:

– $\mathcal{D}(\varepsilon) = (s_0, \boldsymbol{u}_0, (s_0, \mathfrak{f}_0))$; $\mathcal{D}(1) = ((s_0, \mathfrak{f}_0), (\mathfrak{g}^1_1, s^1_1), \ldots, (\mathfrak{g}^1_{\alpha_1}, s^1_{\alpha_1}))$.
– $\ldots$
– $\mathcal{D}(1k_1 1 \cdots k_i) = ((\mathfrak{g}^i_{k_i}, s^i_{k_i}), \boldsymbol{u}_i, (s^i_{k_i}, \mathfrak{f}_i))$.
– $\mathcal{D}(1k_1 1 \cdots k_i 1) = ((s^i_{k_i}, \mathfrak{f}_i), (\mathfrak{g}^{i+1}_1, s^{i+1}_1), \ldots, (\mathfrak{g}^{i+1}_{\alpha_{i+1}}, s^{i+1}_{\alpha_{i+1}}))$.
– $\ldots$
– $\mathcal{D}(1k_1 1 \cdots k_{\beta-1}) = ((\mathfrak{g}^{\beta-1}_{k_i}, s^{\beta-1}_{k_{\beta-1}}), \boldsymbol{u}_{\beta-1}, (s^{\beta-1}_{k_{\beta-1}}, \mathfrak{f}_{\beta-1}))$.
– $\mathcal{D}(1k_1 1 \cdots k_{\beta-1}1) = ((s^{\beta-1}_{k_{\beta-1}}, \mathfrak{f}_{\beta-1}), (\mathfrak{g}^\beta_1, s^\beta_1), \ldots, (\mathfrak{g}^\beta_{\alpha_\beta}, s^\beta_{\alpha_\beta}))$.
– $\cdots$

Then,

$$\mathtt{ext}(\mathfrak{w}, F_A) \stackrel{\text{def}}{=} s_0 \xrightarrow{\boldsymbol{u}_0} (s_0, \mathfrak{f}_0) \rightarrow (\mathfrak{g}^1_{k_1}, s^1_{k_1}) \xrightarrow{\boldsymbol{u}_1} (s^1_{k_1}, \mathfrak{f}_1) \rightarrow (\mathfrak{g}^2_{k_2}, s^2_{k_2}) \xrightarrow{\boldsymbol{u}_2} (s^2_{k_2}, \mathfrak{f}_2) \rightarrow (\mathfrak{g}^3_{k_3}, s^3_{k_3}) \cdots$$

Lemma 3.

(I) *For every maximal computation λ starting at s and respecting F_A, there is maximal branch $\mathfrak{w}$ in $\mathcal{D}$ such that* $\mathtt{ext}(\lambda, F_A) = \mathtt{ext}(\mathfrak{w}, F_A)$.

(II) *For every maximal branch $\mathfrak{w}$ in $\mathcal{D}$, there is a maximal computation λ starting at s and respecting F_A such that* $\mathtt{ext}(\mathfrak{w}, F_A) = \mathtt{ext}(\lambda, F_A)$.

The proof is by an easy verification but this is the place where the proponent restriction condition is essential to perform the reduction and to get its correctness.

Theorem 1. *There is a $\boldsymbol{b}$-consistent strategy w.r.t. s in $\mathfrak{M}$ iff there is a proof in $\mathcal{A}_{\mathfrak{M},A,s}$ whose root is labelled by $(s, \boldsymbol{b})$ and every maximal branch is infinite.*

Proof. First suppose that there is a $\boldsymbol{b}$-consistent strategy F_A w.r.t. s in $\mathfrak{M}$. Let us consider the structures $(\mathfrak{T}_{F_A}, \mathfrak{R}, \mathfrak{L})$ and $\mathcal{D} : \mathfrak{T} \rightarrow (R_1 \cup R_2)$ as defined above. By Lemma 1, there is a unique derivation $\hat{\mathcal{D}}$ of $\mathcal{A}_{\mathfrak{M},A,s}$ based on $\mathcal{D}$ such that $\hat{\mathcal{D}}(\varepsilon) = (q, \boldsymbol{b})$. It remains to prove that $\hat{\mathcal{D}}$ is indeed a proof. Let $\mathfrak{w} = 1k_1 1k_2 1 \cdots$ be a maximal branch of $\hat{\mathcal{D}}$. We have:

– $\hat{\mathcal{D}}(\varepsilon) = (s_0, \boldsymbol{b})$.
– $\hat{\mathcal{D}}(1) = ((s_0, \mathfrak{f}_0), \boldsymbol{b} + \boldsymbol{u}_0)$ with $\mathcal{D}(\varepsilon) = (s_0, \boldsymbol{u}_0, (s_0, \mathfrak{f}_0))$.
– $\ldots$

– $\hat{\mathcal{D}}(1k_1 1 \cdots k_i) = ((\mathfrak{g}^i_{k_i}, s^i_{k_i}), \boldsymbol{b} + \sum_{j=1}^{i-1} \boldsymbol{u}_j)$ with

$$\mathcal{D}(1k_1 1 \cdots k_{i-1} 1) = ((s^{i-1}_{k_{i-1}}, \mathfrak{f}_{i-1}), (\mathfrak{g}^i_1, s^i_1), \ldots, (\mathfrak{g}^i_{\alpha_i}, s^i_{\alpha_i})).$$

– $\hat{\mathcal{D}}(1k_1 1 \cdots k_i 1) = ((s^i_{k_i}, \mathfrak{f}_i), \boldsymbol{b} + \sum_{j=1}^{i} \boldsymbol{u}_j)$ with $\mathcal{D}(1k_1 1 \cdots k_i) = ((\mathfrak{g}^i_{k_i}, s^i_{k_i}), \boldsymbol{u}_i, (s^i_{k_i}, \mathfrak{f}_i))$.
– $\cdots$

By Lemma 3, there is a maximal computation λ starting at s and respecting F_A such that $\mathtt{ext}(\mathfrak{w}, F_A) = \mathtt{ext}(\lambda, F_A)$. Since F_A is a $\boldsymbol{b}$-consistent strategy, λ is $\boldsymbol{b}$-consistent and therefore for all $i \geq 0$, we have $\mathbf{0} \leq \boldsymbol{b} + \sum_{j=1}^{i-1} \boldsymbol{u}_j$, which implies that $\hat{\mathcal{D}}$ is a proof.

For the proof of the other direction, assuming that there is a proof $\hat{\mathcal{D}}$ whose root is labelled by $(s, \boldsymbol{b})$ and every maximal branch is infinite, we can extract from the underlying derivation $\mathcal{D}$ a strategy F_A (see a similar construction in the proof of Theorem 2). Lemma 3 and the fact that $\hat{\mathcal{D}}$ is admissible entail that F_A is $\boldsymbol{b}$-consistent strategy w.r.t. s (details are omitted).

Transitions in $\mathfrak{M}$ can be defined as triples $(s', \mathfrak{g}, s'')$ such that $\delta(s', \mathfrak{g}) = s''$. A transition is also denoted by the expression $s' \xrightarrow{\mathfrak{g}} s''$. The set of transitions of $\mathfrak{M}$ is written below $\Sigma_{\mathfrak{M}}$ and it is understood as a finite alphabet when $\mathfrak{M}$ is finite. An infinite computation $\lambda = s_0 \xrightarrow{\mathfrak{g}_1} s_1 \xrightarrow{\mathfrak{g}_2} s_2 \ldots$ can be equivalently represented by the ω-word in $\Sigma^\omega_{\mathfrak{M}}$ (with contiguous transitions)

$$(s_0 \xrightarrow{\mathfrak{g}_1} s_1) \cdot (s_1 \xrightarrow{\mathfrak{g}_2} s_2) \cdot (s_2 \xrightarrow{\mathfrak{g}_3} s_3) \cdots$$

An ω-word $\mathfrak{w} \in \Sigma^\omega_{\mathfrak{M}}$ is said *to be with contiguous transitions* whenever at any position, the second state of the transition is equal to the first state of the next position.

Given an infinite branch of the proof corresponding to the extended computation

$$s \xrightarrow{\boldsymbol{u}_0} (s, \mathfrak{f}_0) \rightarrow (\mathfrak{g}^1_{k_1}, s^1_{k_1}) \xrightarrow{\boldsymbol{u}_1} (s^1_{k_1}, \mathfrak{f}_1) \rightarrow (\mathfrak{g}^2_{k_2}, s^2_{k_2}) \xrightarrow{\boldsymbol{u}_2} (s^2_{k_2}, \mathfrak{f}_2) \rightarrow (\mathfrak{g}^3_{k_3}, s^3_{k_3}) \cdots$$

its $\Sigma_{\mathfrak{M}}$-projection is defined as the sequence

$$(s \xrightarrow{\mathfrak{g}^1_{k_1}} s^1_{k_1}) \cdot (s^1_{k_1} \xrightarrow{\mathfrak{g}^2_{k_2}} s^2_{k_2}) \cdot (s^2_{k_2} \xrightarrow{\mathfrak{g}^3_{k_3}} s^3_{k_3}) \cdots$$

Lemma 4. *Let* $\mathrm{L} \subseteq \Sigma^\omega_{\mathfrak{M}}$ *and* $\boldsymbol{b} \in (\mathbb{N} \cup \{\omega\})^r$. *The statements below are equivalent.*

1. *There is a* $\boldsymbol{b}$*-consistent strategy* F_A *w.r.t.* s *in* $\mathfrak{M}$ *such that the set of computations* $\mathtt{out}(s, F_A)$ *is included in* L.
2. *There is a proof in* $\mathcal{A}_{\mathfrak{M},A,s}$ *whose root is labelled by* $(s, \boldsymbol{b})$*, every maximal branch is infinite and its* $\Sigma_{\mathfrak{M}}$*-projection belongs to* L.

Lemma 4 is a consequence of Theorem 1 and Lemma 3 and it is instrumental to establish formal correspondences between $\mathfrak{M}$ and $\mathcal{A}_{\mathfrak{M},A,s}$. Now, the main point is to determine classes of languages for which decidability can be regained by using only the decidability (and complexity characterisation) of the state reachability and non-termination problems for AVASS.

Given $S' \subseteq S$ with $s \in S'$, let $\mathrm{L}_{S'}$ be the set of all ω-words such that the transitions use only states in S'. Let us define $\mathcal{A}^{S'}_{\mathfrak{M},A,s}$ as a restriction of $\mathcal{A}_{\mathfrak{M},A,s}$ in which the opponent has no way to go out of S'. Alternatively, $\mathcal{A}^{S'}_{\mathfrak{M},A,s}$ can be understood as the restriction of $\mathcal{A}_{\mathfrak{M},A,s}$ to rules that only involve states in S' (assuming that s is already in S'). We define $\mathcal{A}^{S'}_{\mathfrak{M},A,s} \stackrel{\text{def}}{=} (Q, r, R_1, R_2)$ as follows:

$$Q \stackrel{\text{def}}{=} \{s\} \cup \{(s', \mathfrak{f}) \mid s' \in S',\ \mathfrak{f} \in D_A(s')\} \cup \{(\mathfrak{g}, s') \mid s', s'' \in S',\ \mathfrak{g} \in D(s''),\ \delta(s'', \mathfrak{g}) = s'\}.$$

- The set of unary rules R_1 contains the following elements:
  - For all $\mathfrak{f} \in D_A(s)$, $(s, \mathtt{cost}_A(s, \mathfrak{f}), (s, \mathfrak{f}))$.
  - For all $(\mathfrak{g}, s') \in Q$, for all $\mathfrak{f} \in D_A(s')$, $((\mathfrak{g}, s'), \mathtt{cost}_A(s', \mathfrak{f}), (s', \mathfrak{f}))$.
- The set of fork rules R_2 contains the following elements.
  - For all $(s', \mathfrak{f}) \in Q$, let $\{(\mathfrak{g}_1, s_1), \ldots, (\mathfrak{g}_\alpha, s_\alpha)\} = \{(\mathfrak{g}, s'') \in S \mid s'' \in \delta(s', \mathfrak{g}),\ \mathfrak{g} \in D(s'),\ \mathfrak{f} \sqsubseteq \mathfrak{g}\}$.
    If $\{s_1, \ldots, s_\alpha\} \subseteq S'$, then we add the α-ary fork rule $((s', \mathfrak{f}), (\mathfrak{g}_1, s_1), \ldots, (\mathfrak{g}_\alpha, s_\alpha))$.
  So, there is at most one fork rule starting by $(s', \mathfrak{f})$ (possibly zero).

Lemma 5. *Assuming that $s \in S'$, the statements below are equivalent.*

1. *There is a $\boldsymbol{b}$-consistent strategy F_A w.r.t. s in $\mathfrak{M}$ such that the set of computations $\mathtt{out}(s, F_A)$ only visit states in S'.*
2. *There is a proof in $\mathcal{A}^{S'}_{\mathfrak{M},A,s}$ whose root is labelled by $(s, \boldsymbol{b})$ and every maximal branch is infinite (a positive instance of the non-termination problem for AVASS).*

Proof. Let $\mathcal{A}_{\mathfrak{M},A,s} = (Q, r, R_1, R_2)$ and $\mathcal{A}^{S'}_{\mathfrak{M},A,s} = (Q', r, R'_1, R'_2)$. By construction, we have $S' \subseteq S$, $R'_1 \subseteq R_1$ and $R'_2 \subseteq R_2$. We write $\Sigma'_{\mathfrak{M}}$ to denote the alphabet $\{s_1 \xrightarrow{\mathfrak{g}} s_2 : s_1, s_2 \in S' \text{ and } \delta(s_1, \mathfrak{g}) = s_2\}$ and L to denote the ω-regular language in $(\Sigma'_{\mathfrak{M}})^\omega$ made of infinite sequences of contiguous transitions such that only states in S' can occur.

(1) $\rightarrow$ (2). Suppose there is a $\boldsymbol{b}$-consistent strategy F_A w.r.t. s in $\mathfrak{M}$ such that the set of computations $\mathtt{out}(s, F_A)$ only visit states in S', which amounts to have the set of computations included in L. By Lemma 4, there is a proof in $\mathcal{A}_{\mathfrak{M},A,s}$ whose root is labelled by $(s, \boldsymbol{b})$, every maximal branch is infinite and it belongs to L. Since $s \in S'$, all the rules in $R'_1 \cup R'_2$ only involve states in S' and the above proof only visits such states, we get that there is a proof in $\mathcal{A}^{S'}_{\mathfrak{M},A,s}$ whose root is labelled by $(s, \boldsymbol{b})$ and every maximal branch is infinite.

(2) $\rightarrow$ (1). Suppose that there is a proof in $\mathcal{A}^{S'}_{\mathfrak{M},A,s}$ whose root is labelled by $(s, \boldsymbol{b})$ and every maximal branch is infinite. Since $\mathcal{A}^{S'}_{\mathfrak{M},A,s}$ is defined as a restriction of $\mathcal{A}_{\mathfrak{M},A,s}$ with $S' \subseteq S$, $R'_1 \subseteq R_1$ and $R'_2 \subseteq R_2$, there is also a proof in $\mathcal{A}_{\mathfrak{M},A,s}$ whose root is labelled by $(s, \boldsymbol{b})$, every maximal branch is infinite and only states in S' are visited. Equivalently, there is a proof in $\mathcal{A}_{\mathfrak{M},A,s}$ whose root is labelled by $(s, \boldsymbol{b})$, every maximal branch is infinite and it belongs to L. By Lemma 4, there is a $\boldsymbol{b}$-consistent strategy w.r.t. s in $\mathfrak{M}$ such that the set of computations $\mathtt{out}(s, F_A)$ is included in L, whence only states in S' are visited.

Given $S_1, S_2 \subseteq S$ with $s \in S_1 \cup S_2$, let L_{S_1,S_2} be the set of all ω-words with contiguous transitions such that the projection under S belongs to $S_1^* \cdot S_2 \cdot S^\omega$. Typically, the projection of $(s_0 \xrightarrow{\mathfrak{g}_1} s_1) \cdot (s_1 \xrightarrow{\mathfrak{g}_2} s_2) \cdot (s_2 \xrightarrow{\mathfrak{g}_3} s_3) \cdots$ under S is precisely $s_0 s_1 s_2 s_3 \cdots$.

Lemma 6. *The statements below are equivalent:*

1. *There is a $\boldsymbol{b}$-consistent strategy w.r.t. s in $\mathfrak{M}$ such that* $\mathtt{out}(s, F_A) \subseteq \mathrm{L}_{S_1,S_2}$.
2. *There is a finite proof in $\mathcal{A}^{S_1 \cup S_2}_{\mathfrak{M},A,s}$ whose root is labelled by $(s, \boldsymbol{b})$ and each leaf contains a control state in $\{(\mathfrak{g}, s') \in Q \mid s' \in S_2\} \cup \{s' \in Q \mid s' = s,\ s \in S_2\}$. (a positive instance of the state reachability problem for AVASS).*

The set $\{s' \in Q \mid s' = s,\ s \in S_2\}$ may look a bit strange: basically it takes the value $\{s\}$ if $s \in S_2$ and $\emptyset$ otherwise. The proof of Lemma 6 below makes an essential use of the property in resource-bounded concurrent game structures that requires that $\mathtt{idle} \in \mathtt{act}(a, s)$ for all agents and states.

Proof. The proof follows the lines of the proof of Lemma 5 except that we need to relate finite proofs to infinite ones and for that purpose we may take advantage of the presence of the idle action in the game structures.

Let $\mathcal{A}_{\mathfrak{M},A,s} = (Q, r, R_1, R_2)$ and $\mathcal{A}^{S_1 \cup S_2}_{\mathfrak{M},A,s} = (Q', r, R'_1, R'_2)$. By construction, we have $S_1 \cup S_2 \subseteq S$, $R'_1 \subseteq R_1$ and $R'_2 \subseteq R_2$.

(1) $\rightarrow$ (2). Suppose there is a $\boldsymbol{b}$-consistent strategy F_A w.r.t. s in $\mathfrak{M}$ such that the computations in $\mathtt{out}(s, F_A)$ visit a state in S_1 until a state in S_2 is visited, which amounts to have the set of computations included in L_{S_1,S_2}. By Lemma 4, there is a proof $\hat{\mathcal{D}}$ in $\mathcal{A}_{\mathfrak{M},A,s}$ whose root is labelled by $(s, \boldsymbol{b})$, every maximal branch is infinite and it belongs to L_{S_1,S_2}. Let $\hat{\mathcal{D}}'$ be the finite proof obtained from $\hat{\mathcal{D}}$ by pruning any subtree as soon as a node is labelled by a control state in S_2. Existence of such a finite proof is guaranteed by König's Lemma. It is easy to check that $\hat{\mathcal{D}}'$ is a finite proof in $\mathcal{A}^{S_1 \cup S_2}_{\mathfrak{M},A,s}$ whose root is labelled by $(s, \boldsymbol{b})$ and each leaf contains a control state in $\{(\mathfrak{g}, s') \in Q \mid s' \in S_2\} \cup \{s' \in Q \mid s' = s,\ s \in S_2\}$.

(2) $\rightarrow$ (1). Suppose that there is a finite proof $\hat{\mathcal{D}}$ in $\mathcal{A}^{S_1 \cup S_2}_{\mathfrak{M},A,s}$ whose root is labelled by $(s, \boldsymbol{b})$ and each leaf contains a control state in $\{(\mathfrak{g}, s') \in Q \mid s' \in S_2\} \cup \{s' \in Q \mid s' = s,\ s \in S_2\}$. One can extend $\hat{\mathcal{D}}$ in order to obtain an infinite proof $\hat{\mathcal{D}}'$ such that every maximal branch is infinite and it belongs to L_{S_1,S_2}. Indeed, any leaf labelled by the control state $(\mathfrak{g}, s')$ is further extended by application of the unary rule $(\mathfrak{g}, s') \xrightarrow{\mathbf{0}} (s', \mathfrak{f})$ where $\mathfrak{f}$ is the idle joint action (with the control state s, a similar method applies). Similarly, any leaf labelled by the control state $(s', \mathfrak{f})$ is further extended by application of the unique fork rule starting by $(s', \mathfrak{f})$. It is easy to check that this leads not only to a derivation but also to a proof because the extension only deals with the harmless update vector $\mathbf{0}$. By Lemma 4, there is a $\boldsymbol{b}$-consistent strategy w.r.t. s in $\mathfrak{M}$ such that the set of computations $\mathtt{out}(s, F_A)$ is included in L_{S_1,S_2}.

4.2 2EXPTIME-hardness

In this section, we show a 2EXPTIME-hardness result by reduction from the state reachability problem for AVASS. It is worth noting that the proof still works if no distinguished idle action is required in the game structures or if `act` may return an empty set of actions, see e.g. Definition 1. This improves the EXPSPACE-hardness result in [ALNR15].

Theorem 2. *The model-checking problem for RB±ATL is* 2EXPTIME*-hard.*

Proof. The proof is by reduction from the state reachability problem for AVASS (see Proposition 2 or [CS14, Theorem 4.1]). It can be divided in three main parts.

1. We consider a restriction of the state reachability problem for AVASS that remains 2EXPTIME-hard but that simplifies the definitions in the second part. Roughly speaking, the set of control states can be divided in two disjoint sets, one from which starts unary rules and the other one from which starts the fork rules.
2. We define the reduction from the restriction by taking care of the peculiarities of the resource-bounded concurrent game structures but essentially we follow ideas that are similar to those to prove [ALNR15, Lemma 6].
3. We establish the correctness of the reduction.

(1) Given an instance $\mathcal{A} = (Q, r, R_1, R_2)$, q_0, q_f of the state reachability problem, we further assume that there is a partition $Q = Q_1 \uplus Q_2$ such that

- $q_0, q_f \in Q_1$,
- $R_1 \subseteq Q_1 \times \mathbb{Z}^r \times Q_2$ and $R_2 \subseteq Q_2 \times Q_1 \times Q_1$,
- there is no rule starting by q_f and there is a unary rule starting by q_0.

The strict alternation between control states in Q_1 and those in Q_2 can be obtained by duplicating the number of control states (in case a control state can start simultaneously a unary rule and a fork rule) and by adding new intermediate rules to enforce the alternation. In order to have no rule from q_f, it is sufficient to duplicate it and to reach q_f only for the last visit. The details follow.

Let $\mathcal{A} = (Q, r, R_1, R_2)$ be an alternating VASS. Without any loss of generality, we can assume that no rule starts by q_f. Otherwise, we can introduce a new control state q_f^{new} that behaves almost as q_f: copy all the rules where q_f occurs in second or in third position by replacing q_f by q_f^{new} but no rule starting by q_f is copied (details are omitted). We can guarantee that ($\star$) there is a finite proof whose root is equal to $(q_0, \mathbf{0})$ and each leaf belongs to $\{q_f\} \times \mathbb{N}^r$ iff with the new AVASS there is a finite proof whose root is equal to $(q_0, \mathbf{0})$ and each leaf belongs to $\{q_f^{\text{new}}\} \times \mathbb{N}^r$. Similarly, without any loss of generality, we can assume that there is a rule in R_1 that starts by q_0. Otherwise, we add the dummy unary rule $(q_0, \mathbf{0}, q_0)$.

Let us build the alternating VASS $\mathcal{A}' = (Q', r, R_1', R_2')$ verifying the above conditions with $Q' = Q_1' \uplus Q_2'$ such that ($\star$) iff there is a finite proof whose root is equal to $((q_0, 1), \mathbf{0})$ and each leaf belongs to $\{(q_f, 1)\} \times \mathbb{N}^r$. The set Q' is a subset of $\{1, 2\} \times Q$ defined by the clauses below plus auxiliary states introduced with the definition for rules:

- $(q, 1) \in Q' \overset{\text{def}}{\Leftrightarrow}$ there is a rule in R_1 that starts by q or $q = q_f$. So $(q_0, 1) \in Q'$.
- $(q, 2) \in Q' \overset{\text{def}}{\Leftrightarrow}$ there is a rule in R_2 that starts by q.
- $Q_1' \supseteq \{(q, i) \in Q' \mid i = 1\}$ and $Q_2' \supseteq \{(q, i) \in Q' \mid i = 2\}$. Obviously, $(q_0, 1), (q_f, 1) \in Q_1'$.

Now, it is time to define the sets of rules R_1' and R_2'.

- For all $q \xrightarrow{u} q' \in R_1$ such that $(q', 2) \in Q'$, we add the rule $(q, 1) \xrightarrow{u} (q', 2)$ to R_1'.

– For all $(q_1, q_2, q_3) \in R_2$ such that $(q_2, 1), (q_3, 1) \in Q'$, we add the new fork rule $((q_1, 2), (q_2, 1), (q_3, 1))$ to R'_2.
– For all $\mathfrak{r} = q \xrightarrow{\boldsymbol{u}} q' \in R_1$ such that $(q', 1) \in Q'$, we add the rules below: $(q, 1) \xrightarrow{\boldsymbol{u}} q^{\text{new}} \in R'_1$ (q^{new} is new and depends on $\mathfrak{r}$) and $(q^{\text{new}}, (q', 1), (q', 1)) \in R'_2$. Moreover, $q^{\text{new}} \in Q'_2$. This amounts to add an intermediate fork rule leading twice to $(q', 1)$ in order to guarantee that $R'_1 \subseteq Q'_1 \times Q'_2$.
– For all $\mathfrak{r} = (q_1, q_2, q_3) \in R_2$ such that either $(q_2, 2) \in Q'$ or $(q_3, 2) \in Q'$. If $(q_2, 2), (q_3, 2) \in Q'$, then we add the rules below:
  • $((q_1, 2), q_2^{\text{new}}, q_3^{\text{new}}) \in R'_2$ where q_2^{new} and q_3^{new} are new and depend on the fork rule $\mathfrak{r}$. Moreover, these two new control states belong to Q'_1.
  • $q_2^{\text{new}} \xrightarrow{\mathbf{0}} (q_2, 2)$ and $q_3^{\text{new}} \xrightarrow{\mathbf{0}} (q_3, 2)$ belong to R'_1.

  Again, we add an intermediate unary rules in order to guarantee that $R'_2 \subseteq Q'_2 \times Q'_1 \times Q'_1$. If $(q_2, 2)$, $(q_3, 1)$ belong to Q' or if $(q_2, 1)$, $(q_3, 2)$ belong to Q', the contruction can be easily adapted.

One can show that $\mathcal{A}' = (Q', r, R'_1, R'_2)$ satisfies the above assumption and ($\star$) iff there is a finite proof whose root is equal to $((q_0, 1), \mathbf{0})$ and each leaf belongs to $\{(q_f, 1)\} \times \mathbb{N}^r$.

(2) Given a instance $\mathcal{A} = (Q, r, R_1, R_2)$, q_0 and q_f with the above-mentioned restriction, let us build the game structure $\mathfrak{M} = (Agt, S, Act, r, \texttt{act}, \texttt{cost}, \delta, L)$ with $Q_1 \subseteq S$ such that $\mathfrak{M}, q_0 \models \langle\langle \{1\}^{\mathbf{0}} \rangle\rangle \top \mathcal{U} q_f$ iff there is a finite proof of AVASS whose root is equal to $(q_0, \mathbf{0})$ and each leaf belongs to $\{q_f\} \times \mathbb{N}^r$. Here, q_f is also understood as a propositional variable. Let us complete the definition of $\mathfrak{M}$.

– $Agt \stackrel{\text{def}}{=} \{1, 2\}$.
– A pair of rules $(\mathfrak{r}_1, \mathfrak{r}_2) \in R_1 \times R_2$ is *connected* iff the last control state of $\mathfrak{r}_1$ is equal to the first control state of $\mathfrak{r}_2$. The set of actions Act is equal to the set of connected pairs of rules plus the action `idle` and its twin action `idle'`.
– $S = Q_1 \uplus \{s_{\text{bad}}\}$.
– For each control state q in Q_1, $\texttt{act}(1, q)$ is the set of connected pairs of rules whose unary rule starts by q plus the idle action. So the agent 1 can choose a unary rule immediately followed by a fork rule. With such a definition, $\texttt{act}(1, q_f)$ is restricted to $\{\texttt{idle}\}$ because no rule starts by q_f in $\mathcal{A}$.
– As far as the agent 2 is concerned, for all $q \in Q_1$, $\texttt{act}(2, q) \stackrel{\text{def}}{=} \{\texttt{idle}, \texttt{idle'}\}$. So the agent 2 can perform two actions with no effect on resources, which amounts to simulate the effects of fork rules.
– Nothing else can be done from the bad state s_{bad}: $\texttt{act}(1, s_{\text{bad}}) \stackrel{\text{def}}{=} \texttt{act}(2, s_{\text{bad}}) \stackrel{\text{def}}{=} \{\texttt{idle}\}$.
– Let us now define the cost function: the cost of the action $(\mathfrak{r}_1, \mathfrak{r}_2)$ is simply the update vector of the unary rule $\mathfrak{r}_1$. Formally, for all $q \in Q_1$, we have $\texttt{cost}(q, 1, (\mathfrak{r}_1, \mathfrak{r}_2)) \stackrel{\text{def}}{=} \boldsymbol{u}$ when $\mathfrak{r}_1 = (q, \boldsymbol{u}, q')$ for some q'.

  Furthermore, $\texttt{cost}(q, a, \texttt{idle}) \stackrel{\text{def}}{=} \texttt{cost}(s_{\text{bad}}, a, \texttt{idle}) \stackrel{\text{def}}{=} \mathbf{0}$ for all $a \in \{1, 2\}$ as expected. Finally $\texttt{cost}(q, 2, \texttt{idle'}) \stackrel{\text{def}}{=} \mathbf{0}$.
– When δ can be defined, we have
  • $\delta(q_f, \mathfrak{f}) \stackrel{\text{def}}{=} q_f$; $\delta(s_{\text{bad}}, \mathfrak{f}) \stackrel{\text{def}}{=} s_{\text{bad}}$ ($\mathfrak{f}$ is necessarily the constant map `idle`).

- Whenever $q \in (Q_1 \setminus \{q_f\})$, $\mathfrak{f}(1) = (\mathfrak{r}_1, \mathfrak{r}_2)$ with $\mathfrak{r}_1$ starting by q and $\mathfrak{r}_2 = (q_{inter}, q', q'')$,

$$\delta(q, \mathfrak{f}) \stackrel{\text{def}}{=} \begin{cases} q' & \text{if } \mathfrak{f}(2) = \texttt{idle} \\ q'' & \text{otherwise, i.e. } \mathfrak{f}(2) = \texttt{idle}'. \end{cases}$$

- $\delta(q, \mathfrak{f}) = s_{\text{bad}}$ whenever $\mathfrak{f}(1) = \texttt{idle}$ for all $q \in Q_1 \setminus \{q_f\}$.

– There is a unique propositional variable q_f and $L(q_f) \stackrel{\text{def}}{=} \{q_f\}$.

(3) Without any loss of generality, we can assume $q_0 \neq q_f$.

First, suppose that $\mathfrak{M}, q_0 \models \langle\langle\{1\}^{\mathbf{0}}\rangle\rangle \top \boldsymbol{\mathcal{U}} q_f$. So, there exists a $\mathbf{0}$-consistent strategy $F_{\{1\}}$ such that for all $\lambda = s_0 \stackrel{\mathfrak{g}_0}{\rightarrow} s_1 \ldots \in \texttt{out}(q_0, F_{\{1\}})$, there is $i \geq 0$ such that $q_i = q_f$ (so by construction of $\mathfrak{M}$, for all $j \geq i$, we have $q_j = q_f$ and s_{bad} does not occur in λ). Since $\{1\}$ is a singleton set, below we assume that $F_{\{1\}}$ returns an action for the agent 1 (instead of returning a joint action with respect to the unique agent 1) and $D_{\{1\}}(q)$ is viewed as an action.

From $\mathfrak{M}$, q_0 and $F_{\{1\}}$, let $(\mathfrak{T}_{F_{\{1\}}}, \mathfrak{R}, \mathfrak{L})$ be the labelled transition system as defined in Section 4.1. We have $\mathfrak{T}_{F_{\{1\}}} \subseteq \{1,2\}^*$, $\mathfrak{L} : \mathfrak{T}_{F_{\{1\}}} \rightarrow Q_1$ (because s_{bad} does not occur) and $\mathfrak{R}$ is partial map $\mathfrak{T}_{F_{\{1\}}} \times \mathfrak{T}_{F_{\{1\}}} \rightarrow \bigcup_{q \in Q_1} D(q)$. Note the specific structure of $(\mathfrak{T}_{F_{\{1\}}}, \mathfrak{R}, \mathfrak{L})$:

– For all $\mathfrak{w} \in \mathfrak{T}_{F_{\{1\}}}$ such that $\mathfrak{L}(\mathfrak{w}) = q_f$, $\mathfrak{w}{\cdot}1$ is the unique successor of $\mathfrak{w}$, $\mathfrak{L}(\mathfrak{w}{\cdot}1) = q_f$ and $\mathfrak{R}(\mathfrak{w}, \mathfrak{w} \cdot 1)$ is the constant joint action equal to $\texttt{idle}$.
– For all $\mathfrak{w} \in \mathfrak{T}_{F_{\{1\}}}$ such that $\mathfrak{w}$ has exactly two successors $\mathfrak{w} \cdot 1$ and $\mathfrak{w} \cdot 2$, there are $q_{\text{inter}} \in Q$ and $\boldsymbol{u} \in \mathbb{Z}^r$ such that

$$\mathfrak{R}(\mathfrak{w}, \mathfrak{w} \cdot 1) = (((\mathfrak{L}(\mathfrak{w}), \boldsymbol{u}, q_{\text{inter}}), (q_{\text{inter}}, \mathfrak{L}(\mathfrak{w} \cdot 1), \mathfrak{L}(\mathfrak{w} \cdot 2))), \texttt{idle})$$

$$\mathfrak{R}(\mathfrak{w}, \mathfrak{w} \cdot 2) = (((\mathfrak{L}(\mathfrak{w}), \boldsymbol{u}, q_{\text{inter}}), (q_{\text{inter}}, \mathfrak{L}(\mathfrak{w} \cdot 1), \mathfrak{L}(\mathfrak{w} \cdot 2))), \texttt{idle}')$$

Now, let us build a finite derivation skeleton $\mathcal{D} : \mathfrak{T} \rightarrow (R_1 \cup R_2 \cup \{\bot\})$ such that its unique derivation with root labelled by $(q_0, \mathbf{0})$ is a finite proof with leaves labelled by q_f. The finite tree $\mathfrak{T}$ can be uniquely defined from $\mathfrak{T}_{F_{\{1\}}}$ by using the map $\mathfrak{c} : \{1,2\}^* \rightarrow \{1,2\}^*$ such that $\mathfrak{c}(\mathfrak{w})$ is obtained from $\mathfrak{w}$ by replacing simultaneously every occurrence of 1 by 11 and every occurrence of 2 by 12. So for instance $\mathfrak{c}(\varepsilon) \stackrel{\text{def}}{=} \varepsilon$ and $\mathfrak{c}(12) \stackrel{\text{def}}{=} 1112$. We pose that $\mathfrak{T}$ is the set of words of the form $\mathfrak{c}(\mathfrak{w})$ or $\mathfrak{c}(\mathfrak{w}) \cdot 1$ where $\mathfrak{w} \in \{1,2\}^*$ and there is $\mathfrak{v} \in \mathfrak{T}_{F_{\{1\}}}$ such that $\mathfrak{w}$ is a (non necessarily strict) prefix of $\mathfrak{v}$, $\mathfrak{L}(\mathfrak{v}) = q_f$ and no strict prefix of $\mathfrak{v}$ is labelled by q_f. The derivation skeleton $\mathcal{D}$ is defined as follows. For all $\mathfrak{w} \in \mathfrak{T}_{F_{\{1\}}}$:

– If $\mathfrak{L}(\mathfrak{w}) \neq q_f$ and $\mathfrak{R}(\mathfrak{w}, \mathfrak{w}{\cdot}1) = ((\mathfrak{r}_1, \mathfrak{r}_2), \texttt{idle})$, then $\mathcal{D}(\mathfrak{c}(\mathfrak{w})) = \mathfrak{r}_1$ and $\mathcal{D}(\mathfrak{c}(\mathfrak{w}){\cdot}1) = \mathfrak{r}_2$.
– If $\mathfrak{L}(\mathfrak{w}) = q_f$ and $\mathfrak{c}(\mathfrak{w}) \in \mathfrak{T}$, then $\mathcal{D}(\mathfrak{c}(\mathfrak{w})) = \bot$.

Similarly to Lemma 3, we can show the following properties:

(I) For every computation λ starting at q_0, ending at q_f and visiting q_f only once and respecting $F_{\{1\}}$, there is maximal branch $\mathfrak{w}$ in $\mathcal{D}$ such that $\texttt{ext}(\lambda, F_{\{1\}}) = \texttt{ext}(\mathfrak{w}, F_{\{1\}})$.

(II) For every maximal branch $\mathfrak{w}$ in $\mathcal{D}$, there is a computation λ starting at q_0, ending at q_f and visiting q_f only once and respecting $F_{\{1\}}$ such that $\mathtt{ext}(\mathfrak{w}, F_{\{1\}}) = \mathtt{ext}(\lambda, F_{\{1\}})$.

Consequently, the unique derivation based on $\mathcal{D}$ with root labelled by $(q_0, \mathbf{0})$ is a finite proof with leaves labelled by q_f. Indeed by construction for all $q \in Q_1$, we have $\mathtt{cost}(q, 1, (\mathfrak{r}_1, \mathfrak{r}_2)) = \boldsymbol{u}$ when $\mathfrak{r}_1 = (q, \boldsymbol{u}, q')$ for some q'.

For the converse direction, let us assume the existence of a finite proof $\hat{\mathcal{D}}$ based on the derivation skeleton $\mathcal{D} : \mathfrak{T} \to (R_1 \cup R_2 \cup \{\bot\})$ such that $\hat{\mathcal{D}}(\varepsilon) = (q_0, \mathbf{0})$ and each leaf is labelled by a pair in $\{q_f\} \times \mathbb{N}^r$. Let us define a strategy $F_{\{1\}}$. First, we require the following properties:

- $F_{\{1\}}(q_0) \stackrel{\text{def}}{=} (\mathcal{D}(\varepsilon), \mathcal{D}(1))$. Since $q_0 \neq q_f$, we know that $1 \in \mathfrak{T}$.
- For all the finite computations λ ending by the state q_f (we have $Q_1 \subseteq S$ and $q_f \in Q_1$), $F_{\{1\}}(\lambda) \stackrel{\text{def}}{=} \mathtt{idle}$.

Let $\lambda = q_0 \xrightarrow{\mathfrak{g}_0} q_1 \xrightarrow{\mathfrak{g}_1} q_2 \cdots \xrightarrow{\mathfrak{g}_{n-1}} q_n$ be a finite computation respecting so far $F_{\{1\}}$ and $q_n \neq q_f$ (since this case is already treated above). Below we define $F_{\{1\}}(\lambda)$.

First, we assume that s_{bad} does not occur in λ. Each joint action $\mathfrak{g}_j$ can be written $((\mathfrak{r}_1^j, \mathfrak{r}_2^j), \mathfrak{b}(k^j))$ with $k^j \in \{1, 2\}$ and $\mathfrak{b} : \{1, 2\} \to \{\mathtt{idle}, \mathtt{idle}'\}$ with $\mathfrak{b}(1) = \mathtt{idle}$ and $\mathfrak{b}(2) = \mathtt{idle}$. The derivation skeleton $\mathcal{D}$ verifies the properties below:

- $\mathcal{D}(\varepsilon) = \mathfrak{r}_1^0$ with $\mathfrak{r}_1^0 = (q_0, \boldsymbol{u}_0, q^0_{\text{inter}})$.
- $\mathcal{D}(1) = \mathfrak{r}_2^0$ with $\mathfrak{r}_2^0 = (q^0_{\text{inter}}, q_1^0, q_2^0)$ and $q^0_{k^0} = q_1$.
- $\mathcal{D}(1k^0) = \mathfrak{r}_1^1$ with $\mathfrak{r}_1^1 = (q^0_{k^0}, \boldsymbol{u}_1, q^1_{\text{inter}})$.
- $\mathcal{D}(1k^01) = \mathfrak{r}_2^1$ with $\mathfrak{r}_2^1 = (q^1_{\text{inter}}, q_1^1, q_2^1)$ and $q^1_{k^1} = q_2$.
- $\ldots$
- $\mathcal{D}(1k^0 1 \cdots k^j) = \mathfrak{r}_1^{j+1}$ with $\mathfrak{r}_1^{j+1} = (q_{k^j}, \boldsymbol{u}_{j+1}, q^{j+1}_{\text{inter}})$.
- $\mathcal{D}(1k^0 1 \cdots k^j 1) = \mathfrak{r}_2^{j+1}$ with $\mathfrak{r}_2^{j+1} = (q^{j+1}_{\text{inter}}, q_1^{j+1}, q_2^{j+1})$ and $q^{j+1}_{k^{j+1}} = q_{j+2}$.
- $\ldots$
- $\mathcal{D}(1k^0 1 \cdots k^{n-2}1) = \mathfrak{r}_2^{n-1}$ with $\mathfrak{r}_2^{n-1} = (q^{n-1}_{\text{inter}}, q_1^{n-1}, q_2^{n-1})$ and $q^{n-1}_{k^{n-1}} = q_n$.

Since q_n is different from q_f, $1k^01 \cdots k^{n-2}1k^{n-1}$ and $1k^01 \cdots k^{n-2}1k^{n-1}1$ exist. We pose $F_{\{1\}}(\lambda) \stackrel{\text{def}}{=} (\mathcal{D}(1k^01 \cdots k^{n-2}1k^{n-1}), \mathcal{D}(1k^01 \cdots k^{n-2}1k^{n-1}1))$. Consequently, any extension $q_0 \xrightarrow{\mathfrak{g}_0} q_1 \xrightarrow{\mathfrak{g}_1} q_2 \cdots \xrightarrow{\mathfrak{g}_{n-1}} q_n \xrightarrow{\mathfrak{g}_n} q_{n+1}$ respecting so far $F_{\{1\}}$ verifies the above correspondence with $\mathcal{D}$ and the state s_{bad} cannot be visited. So, the strategy $F_{\{1\}}$ can be soundly defined by using the above development (more formally, a induction hypothesis should be stated and we should prove that after each step, the property is preserved). One can also check that $F_{\{1\}}$ is $\mathbf{0}$-consistent w.r.t q_0 and for all $\lambda = s_0 \xrightarrow{\mathfrak{g}_0} s_1 \ldots \in \mathtt{out}(q_0, F_{\{1\}})$, there is $i \geq 0$ such that $q_i = q_f$. The $\mathbf{0}$-consistency is due to the fact that $\hat{\mathcal{D}}$ is a proof and the reachability condition is a consequence of the fact that every leaf of $\hat{\mathcal{D}}$ is labelled by q_f thanks to the above correspondences between computations respecting $F_{\{1\}}$ and nodes in $\mathfrak{T}$.

The proof above does not require the satisfaction of the proponent restriction condition or the existence of some idle action. It is worth observing that one propositional variable and two agents are sufficient to get 2EXPTIME-hardness.

4.3 2EXPTIME upper bound

The upper bound is established by designing a labelling algorithm as done in [ALNR15] or for standard temporal logics such as CTL and CTL*. The main difference with the work [ALNR15] rests on the fact that the treatment of the cases with strategy modalities is not performed in some ad-hoc fashion using that $(\mathbb{N}^r, \preceq)$ is a well-quasi-ordering by Dickson's Lemma [Dic13] but rather we explicitly call subroutines that solve decision problems on AVASS. Existence of such subroutines is due to [RSB05] for so-called *monotonic games* whereas their complexity upper bounds are due to [JLS15, Theorem 3.4] and [CS14, Theorem 3.1]. Once more, the proof that establishes the 2EXPTIME upper bound can be divided in three main steps:

- to introduce a slight extension of AVASS such that the decision problems remain in 2EXPTIME,
- to show that the cases for the strategy modalities can be faithfully reduced to subroutines for problems on extended AVASS (a consequence of Section 4.1),
- to design the labelling algorithm and to establish complexity upper bound from it.

So first, let us introduce a slight extension of decision problems for AVASS.

Lemma 7. *The following extension of the state reachability and non-termination problems for AVASS remain in* 2EXPTIME*:*

- *Fork rules can be α-ary for any $\alpha \geq 1$ (but there are only a finite amount of them),*
- *Reachability is related to a subset $S_f \subseteq S$ (instead of a singleton set).*
- *The initial configuration is $(q_0, \boldsymbol{b})$ with $\boldsymbol{b} \in \mathbb{N}^r$ instead of the fixed tuple $\mathbf{0}$.*
- *The value ω in $\boldsymbol{b}$ is allowed and absorbes any other value in $\mathbb{Z}$ (a means to ignore components. i.e. to reduce the dimension).*

The rather standard proof consists in using Proposition 2 by simulating non-binary fork by a linear-size gadget made of unary and binary forking rules and by adding binary forking rules from states in S_f to a new single final state.

Proof. The lemma states four ways to extend the decision problems on AVASS either by slightly extending the notion of AVASS or by considering more general inputs for the problems. For each extension, we show how this can be encoded into the state reachability and the non-termination problems on AVASS while dealing with only polynomial-time reductions. The proof of the lemma is then obtained by composition of the adequate reductions (polynomial-time reductions are also known to be closed under compositions) and by invoking Proposition 2 to get the 2EXPTIME upper bound.

- Let $\mathcal{A} = (Q, r, R_1, R_2)$ be an alternating VASS, $q_0, q_f \in Q$ and $\mathfrak{r} = (q_1, \ldots, q_{\alpha+1})$ be an (extended) α-ary rule. If $\alpha = 1$, the rule can be treated as a unary rule with the update vector $\mathbf{0}$ whereas if $\alpha = 2$, it can be treated as a standard binary fork rule. So assume $\alpha \geq 3$. Let R_2' be the following set of binary fork rules derived from $\mathfrak{r}$ where $q_1', \ldots, q_{\alpha-1}'$ are new control states:

$$R_2' = \{(q_1, q_2, q_1')\} \cup \{(q_j', q_{j+1}, q_{j+1}') \mid j \in [1, \alpha - 2]\} \cup \{(q_{\alpha-1}', q_\alpha, q_{\alpha+1})\}.$$

  It is easy to show that the statements below are equivalent:

- there is a finite proof in $(Q, r, R_1, R_2 \uplus \{\mathrm{r}\})$ whose root is equal to $(q_0, \mathbf{0})$ and each leaf belongs to $\{q_f\} \times \mathbb{N}^r$.
- there is a finite proof in $(Q \uplus \{q'_j \mid j \in [1, \alpha - 1]\}, r, R_1, R_2 \uplus R'_2)$ whose root is equal to $(q_0, \mathbf{0})$ and each leaf belongs to $\{q_f\} \times \mathbb{N}^r$.

If there are more than one extended fork rule, we apply the above reduction as many times as necessary, leading eventually to a reduction to an instance of the state reachability problem for AVASS. The same reduction works fine with the non-termination problem.

– Let $\mathcal{A} = (Q, r, R_1, R_2)$, $q_0 \in Q$ and $S_f = \{q_1, \ldots, q_\beta\}$. Let $\mathcal{A}' = (Q \uplus \{q_f^{\mathrm{new}}\}, r, R'_1, R_2)$ be defined from $\mathcal{A}$ such that $R'_1 \stackrel{\text{def}}{=} R_1 \uplus \{q_i \xrightarrow{\mathbf{0}} q_f^{\mathrm{new}} \mid i \in [1, \beta]\}$. It is easy to show that the statements below are equivalent:
  - there is a finite proof in $\mathcal{A}$ whose root is equal to $(q_0, \mathbf{0})$ and each leaf belongs to $S_f \times \mathbb{N}^r$.
  - there is a finite proof in $\mathcal{A}'$ whose root is equal to $(q_0, \mathbf{0})$ and each leaf belongs to $\{q_f^{\mathrm{new}}\} \times \mathbb{N}^r$.

– Let $\mathcal{A} = (Q, r, R_1, R_2)$, $q_0, q_f \in Q$ and $\boldsymbol{b} \in \mathbb{N}^r$ and $\mathcal{A}' = (Q \uplus \{q'_0\}, r, R_1 \uplus \{q'_0 \xrightarrow{\boldsymbol{b}} q_0\}, R_2)$. It is easy to show that the statements below are equivalent:
  - there is a finite proof in $\mathcal{A}$ whose root is equal to $(q_0, \boldsymbol{b})$ and each leaf belongs to $\{q_f\} \times \mathbb{N}^r$.
  - there is a finite proof in $\mathcal{A}'$ whose root is equal to $(q'_0, \mathbf{0})$ and each leaf belongs to $\{q_f\} \times \mathbb{N}^r$.

  Similarly, the statements below are equivalent:
  - there is a proof in $\mathcal{A}$ whose root is equal to $(q_0, \boldsymbol{b})$ and all the maximal branches are infinite.
  - there is a proof in $\mathcal{A}'$ whose root is equal to $(q'_0, \mathbf{0})$ and all the maximal branches are infinite.

– Let $\mathcal{A} = (Q, r, R_1, R_2)$, $q_0, q_f \in Q$ and $\boldsymbol{b} \in (\mathbb{N} \cup \{\omega\})^r$. We are looking for proofs whose root is labelled by $(q_0, \boldsymbol{b})$ and any occurrence of ω in $\boldsymbol{b}$ remains for all the descendant nodes, which amounts to ignore some components.
  Suppose that ω occurs at least once in $\boldsymbol{b}$ and let $\{i_1, \ldots, i_\beta\} \subseteq [1, r]$ be the set of positions where ω occurs in $\boldsymbol{b}$. Let $r' = r - \beta$ and let $j_1 < \cdots < j_{r'}$ be the indices in $[1, r] \setminus \{i_1, \ldots, i_\beta\}$. Let $\mathfrak{f} : [1, r'] \to \{j_1, \cdots, j_{r'}\}$ be the bijection such that $\mathfrak{f}(n) \stackrel{\text{def}}{=} j_n$. We define the alternating VASS $\mathcal{A}' = (Q, r', R'_1, R_2)$ obtained from $\mathcal{A}$ by removing the components in positions in $\{i_1, \ldots, i_\beta\}$. The map $\mathfrak{f}$ is extended to $\bar{\mathfrak{f}} : (\mathbb{Z} \cup \{\omega\})^r \to \mathbb{Z}^{r'}$ such that for all $\boldsymbol{u} \in \mathbb{Z}^r$, for all $n \in [1, r']$ we have $\bar{\mathfrak{f}}(\boldsymbol{u})(n) \stackrel{\text{def}}{=} \boldsymbol{u}(j_n)$. The set of unary rules R'_1 is defined from R_1 as follows:

$$R'_1 \stackrel{\text{def}}{=} \{q \xrightarrow{\bar{\mathfrak{f}}(\boldsymbol{u})} q' \mid q \xrightarrow{\boldsymbol{u}} q' \in R_1\}.$$

  It is easy to show that the statements below are equivalent:
  - there is a finite proof in $\mathcal{A}$ whose root is equal to $(q_0, \boldsymbol{b})$ and each leaf belongs to $\{q_f\} \times \mathbb{N}^r$.
  - there is a finite proof in $\mathcal{A}'$ whose root is equal to $(q_0, \bar{\mathfrak{f}}(\boldsymbol{b}))$ and each leaf belongs to $\{q_f\} \times \mathbb{N}^r$.

The same reduction works fine with the non-termination problem.

Lemma 8 below is related to the satisfaction of a formula with outermost connectice $\langle\langle A^{\boldsymbol{b}}\rangle\rangle \cdot \boldsymbol{\mathcal{U}} \cdot$ and the state reachability problem for AVASS.

Lemma 8. *The statements below are equivalent:*

(I) $\mathfrak{M}, s \models \langle\langle A^{\boldsymbol{b}}\rangle\rangle \phi_1 \boldsymbol{\mathcal{U}} \phi_2$.

(II) *there is a finite proof in* $\mathcal{A}^{S_1 \cup S_2}_{\mathfrak{M},A,s}$ *whose root is equal to* $(s, \boldsymbol{b})$ *and each leaf has a control state in* $\{(\mathfrak{g}, s') \in Q \mid s' \in S_2\} \cup \{s' \in Q \mid s' = s,\ s \in S_2\}$. *with* $S_i = \{s \mid \mathfrak{M}, s \models \phi_i\}$, $i \in \{1, 2\}$.

This is a direct consequence of Lemma 6. Lemma 9 below is related to the satisfaction of a formula with outermost connectice $\langle\langle A^{\boldsymbol{b}}\rangle\rangle \Box$ and the non-termination problem for AVASS.

Lemma 9. *Assuming that* $\mathfrak{M}, s \models \phi_1$, *the statements below are equivalent:*

(I) $\mathfrak{M}, s \models \langle\langle A^{\boldsymbol{b}}\rangle\rangle \Box \phi_1$.

(II) *there is a proof in* $\mathcal{A}^{S'}_{\mathfrak{M},A,s}$ *with* $S' = \{s'' \mid \mathfrak{M}, s'' \models \phi_1\}$ *whose root is equal to* $(s, \boldsymbol{b})$ *and every maximal branch is infinite.*

This is a direct consequence of Lemma 5.

Theorem 3. *The model-checking problem for RB±ATL is in* 2EXPTIME.

Algorithm 1 An algorithm for RB±ATL model checking.

procedure GMC($\mathfrak{M}, \phi$)
  case ϕ **of**
  p: **return** $\{s \in S \mid p \in L(s)\}$
  $\neg\psi$: **return** $S \setminus GMC(\mathfrak{M}, \psi)$
  $\psi_1 \wedge \psi_2$: **return** $GMC(\mathfrak{M}, \psi_1) \cap GMC(\mathfrak{M}, \psi_2)$
  $\langle\langle A^{\boldsymbol{b}}\rangle\rangle \bigcirc \psi$: **return** $\{s \mid \exists\, \mathfrak{f} \in D_A(s), \boldsymbol{0} \preceq \boldsymbol{b} + \mathtt{cost}_A(s, \mathfrak{f}),$ for all $\mathfrak{f} \sqsubseteq \mathfrak{g} \in D(s),\ \delta(s, \mathfrak{g}) \in GMC(\mathfrak{M}, \psi)\}$
  $\langle\langle A^{\boldsymbol{b}}\rangle\rangle \Box \psi$: $S_1 := GMC(\mathfrak{M}, \psi)$
    if $s \in S_1$ **then return** $\{s \mid \mathcal{A}^{S_1}_{\mathfrak{M},A,s}, (s, \boldsymbol{b})$ is non-terminating$\}$ **end if**
    if $s \notin S_1$ **then return** $\emptyset$ **end if**
  $\langle\langle A^{\boldsymbol{b}}\rangle\rangle \psi_1 \boldsymbol{\mathcal{U}} \psi_2$: **return** $\{s \mid \mathcal{A}^{S_1 \cup S_2}_{\mathfrak{M},A,s}, (s, \boldsymbol{b}), S_2'$ is a positive instance of state reachability$\}$ with $S_1 = GMC(\mathfrak{M}, \psi_1)$, $S_2 = GMC(\mathfrak{M}, \psi_2)$, $S_2' = \{(\mathfrak{g}, s') \in Q \mid s' \in S_2\} \cup \{s' \in Q \mid s' = s,\ s \in S_2\}$
  end case
end procedure

Proof. Algorithm 1 is a global model-checking algorithm that takes as input a game structure $\mathfrak{M}$ and a formula ϕ (both built on the same set of agents and with the same amount of resources) and returns the set of states that satisfies the formula. By structural induction, one can show that $GMC(\mathfrak{M}, \psi) = \{s \mid \mathfrak{M}, s \models \psi\}$ by using Lemma 8 and Lemma 9. Since the state reachability and the non-termination problems for extended AVASS are decidable by [RSB05] (and by using also Lemma 7 for the extension). Let us show how the proof by induction works.
Case $\psi = \langle\langle A^{\boldsymbol{b}}\rangle\rangle \Box \psi'$
The statements below are equivalent:

- $\mathfrak{M}, s \models \psi$.
- there is a $\boldsymbol{b}$-strategy F_A w.r.t. s such that the computations in $\mathtt{out}(s, F_A)$ only visit states in $S_1' = \{s' \mid \mathfrak{M}, s' \models \psi'\}$ (by definition of $\models$).
- $s \in S_1'$ and there is a proof in $\mathcal{A}^{S_1'}_{\mathfrak{M},A,s}$ whose root is equal to $(s, \boldsymbol{b})$ and every maximal branch is infinite (by Lemma 9).
- $s \in S_1$ and there is a proof in $\mathcal{A}^{S_1}_{\mathfrak{M},A,s}$ whose root is equal to $(s, \boldsymbol{b})$ and every maximal branch is infinite with $S_1 = GMC(\mathfrak{M}, \psi')$ (by induction hypothesis).
- $s \in S_1'$ and $\mathcal{A}^{S_1}_{\mathfrak{M},A,s}$, $(s, \boldsymbol{b})$ is a positive instance of the non-terminating problem for AVASS (by definition).

Case $\psi = \langle\langle A^{\boldsymbol{b}}\rangle\rangle \psi_1 \boldsymbol{\mathcal{U}} \psi_2$. The (omitted) proof is similar to the previous case by using Lemma 8 instead of Lemma 9.
Case $\psi = \langle\langle A^{\boldsymbol{b}}\rangle\rangle \bigcirc \psi'$
The statements below are equivalent:

- $\mathfrak{M}, s \models \psi$.
- there is a $\boldsymbol{b}$-strategy F_A w.r.t. s such that for all $s_0 \xrightarrow{\mathfrak{g}_0} s_1 \ldots \in \mathtt{out}(s, F_A)$, we have $\mathfrak{M}, s_1 \models \psi'$ (by definition of $\models$).
- there is a $\boldsymbol{b}$-strategy F_A w.r.t. s such that for all $s_0 \xrightarrow{\mathfrak{g}_0} s_1 \ldots \in \mathtt{out}(s, F_A)$, we have $\mathfrak{M}, s_1 \models \psi'$ and for any finite computation extending $s_0 \xrightarrow{\mathfrak{g}_0} s_1$, F_A returns the constant map $\mathtt{idle}$ (thanks to the properties of the action $\mathtt{idle}$).
- there is $\mathfrak{f} \in D_A(s)$ such that (for all $s_1 \in \mathtt{out}(s, \mathfrak{f})$, we have $\mathfrak{M}, s_1 \models \psi'$) and $\mathbf{0} \preceq \boldsymbol{b} + \mathtt{cost}_A(s, \mathfrak{f})$.
- there is $\mathfrak{f} \in D_A(s)$ such that (for all $\mathfrak{g} \sqsupseteq \mathfrak{f}$, we have $\mathfrak{M}, \delta(s, \mathfrak{g}) \models \psi'$) and $\mathbf{0} \preceq \boldsymbol{b} + \mathtt{cost}_A(s, \mathfrak{f})$.
- there is $\mathfrak{f} \in D_A(s)$ such that (for all $\mathfrak{g} \sqsupseteq \mathfrak{f}$, we have $\delta(s, \mathfrak{g}) \in GMC(\mathfrak{M}, \psi')$) and $\mathbf{0} \preceq \boldsymbol{b} + \mathtt{cost}_A(s, \mathfrak{f})$ (by induction hypothesis).

As far as complexity is concerned, $GMC(\mathfrak{M}, \psi)$ can be solved by using a recursion depth that is linear in the size of ψ and the state reachability and the non-termination problems for AVASS can be solved in 2EXPTIME by [JLS15, Theorem 3.4] and [CS14, Theorem 3.1]. Note also the instances of such problems can be built in polynomial-time in the respective sizes of $\mathfrak{M}$ and ϕ. Consequently, the model-checking problem for RB±ATL is in 2EXPTIME.

Corollary 1. *For any fixed $r \geq 1$, the model-checking problem for RB±ATL restricted to at most r resources is in* EXPTIME. *For $r \geq 4$, the problem is* EXPTIME*-hard.*

Again, we use [JLS15, Theorem 3.4] and [CS14, Theorem 3.1] since for a bounded amount of resources, the state reachability and the non-termination problems for AVASS can be solved in EXPTIME. Moreover, if r is fixed but greater than two, then the model-checking problem for RB±ATL restricted to at most r resources is PSPACE-hard since the state reachability for VASS of dimension two is PSPACE-complete [BFG+15]. When $r = 1$, the model-checking problem for RB±ATL restricted to at most one resource is NP-hard since the state reachability for VASS of dimension one is NP-complete [Haa12]. It is still open how to reduce these two complexity gaps.

5 More Path Formulae While Preserving Decidability

In this section, we study the model-checking problem for resource-bounded logics for which the path formulae are arbitrary, i.e. these can be any LTL-like formulae instead of being restricted to path formulae of the form $\Box\psi$, $\bigcirc\psi$ and $\psi_1 \,\mathcal{U}\, \psi_2$ only as done for RB±ATL. For instance, we have already seen that the model-checking problem for RBTL is EXPSPACE-hard and therefore the lower bound also applies to RBTL*. Below, we show that the model-checking problem for RBTL* is not only decidable (a new result) but also in EXPSPACE. The argument for establishing the EXPSPACE upper bound for RBTL and for RBTL* is identical. By contrast, we have seen that the model-checking problem for RB±ATL is 2EXPTIME-complete. Below, we show that the model-checking problem for RB±ATL* is also decidable but without being able to establish an optimal complexity upper bound. Even though the arguments for establishing the respective decidability of RB±ATL and RB±ATL* both rest on the decidability of decision problems for alternating VASS, for the case of the model-checking problem for RB±ATL* we need to invoke the decidability of parity games on alternating VASS, which is stronger than only invoking the decidability of the state reachability and non-termination problems for AVASS. This more complex reduction, that uses also ingredients such as the same expressive power between Büchi automata and deterministic parity automata on ω-words, will be nevertheless rewarding since this will allow us to synthetise concrete values for resource parameters, which is way beyond what can be done today for all the known resource-bounded logics.

5.1 EXPSPACE upper bound for RBTL

The EXPSPACE lower bound for the model-checking problem for RBTL can be matched with the upper bound for RBTL*.

Theorem 4. *The model-checking problem for RBTL* is in* EXPSPACE.

In [BF09], the problem for RBTL is shown decidable by reduction into the reachability problem for VASS. However the best known upper bound for the reachability problem is $\mathbf{F}_{\omega^3}$, belonging to the fast-growing complexity hierarchy [LS15]. Hence, the EXPSPACE upper bound is a substantial improvement, that leads to a complexity characterisation. Moreover, the decidability status of RBTL* was left open in [BF09].

Proof. (sketch) The algorithm to get the EXPSPACE upper bound first computes on which states the subformulae hold before dealing with larger formulae. This is indeed a renaming algorithm. However, there is a caveat: when dealing with subformulae of the form $\langle \boldsymbol{b} \rangle \Psi$ where Ψ is an LTL formula, we are entitled to use the model-checking algorithm for LTL formulae on VASS that is in EXPSPACE [Hab97] (having ω in one component amounts to ignore that position). However, in order to systematically consider such subformulae $\langle \boldsymbol{b} \rangle \Psi$ when the outermost connective is a path quantifier, we need to perform renamings on-the-fly.

Let us provide a simple example with the formula

$$\phi = \langle \boldsymbol{b}_0 \rangle \, \Box\Diamond \, \langle \boldsymbol{b}_1 \rangle \, q \mathcal{U} q'$$

and with an arbitrary model $\mathfrak{M}$. Let us consider some innermost state formula prefixed by a path quantifier, say $\langle \boldsymbol{b}_1 \rangle (q \mathcal{U} q')$ (actually here there is only one such a subformula). With the help of a decision procedure for solving the LTL model-checking problem on VASS, we determine for which control state q'' we have $\mathfrak{M}, q'' \models_{\mathrm{s}} \langle \boldsymbol{b}_1 \rangle (q \mathcal{U} q')$. Say, we obtain the set $\{q_1, \ldots, q_\alpha\}$. Now, in ϕ, we replace $\langle \boldsymbol{b}_1 \rangle (q \mathcal{U} q)$ by $q_1 \vee \cdots \vee q_\alpha$, say we get $\phi_1 = \langle \boldsymbol{b}_0 \rangle \, \Box\Diamond \, (q_1 \vee \cdots \vee q_\alpha)$. So, we have performed a renaming step by replacing a subformula by a disjunction of propositional variables. This process can be repeated until there is no more path quantifiers. So, again we substitute some innermost state formula prefixed by a path quantifier in ϕ_1, say $\langle \boldsymbol{b}_0 \rangle \Box\Diamond \, (q_1 \vee \cdots \vee q_\alpha)$, by a new disjunction of locations (possibly empty) with the help of a decision procedure for solving the LTL model-checking problem on VASS. We obtain the manageable formula

$$\phi_2 = q'_1 \vee \cdots \vee q'_\beta.$$

Since ϕ_2 is a propositional formula, we are done with the renaming process and it is easy to show that for every control state q'', we have $\mathfrak{M}, q'' \models_{\mathrm{s}} \phi$ iff $\mathfrak{M}, q'' \models q'_1 \vee \cdots \vee q'_\beta$. The above example can be easily generalized to any state formula. Note that the number of renamings is bounded by the size of the initial formula and at each step a subroutine is invoked at most card(Q) times and it requires EXPSPACE, whence we obtain the promised EXPSPACE upper bound.

Corollary 2. *For any fixed $r \geq 1$, the model-checking problem for RBTL* restricted to at most r resources is in* PSPACE.

The PSPACE upper bound is then a consequence of [Hab97, Theorem 4.1]. Again, if r is fixed but greater than two, then the model-checking problem for RBTL* restricted to at most r resources is PSPACE-hard since the state reachability problem for VASS of dimension two is PSPACE-complete [BFG$^+$15]. When $r = 1$, the model-checking problem for RBTL* restricted to at most one resource is NP-hard since the state reachability for VASS of dimension one is NP-complete [Haa12].

5.2 Decidability of RB±ATL*

Below, we introduce RB±ATL*, an extension of RB±ATL in which the path formulae are unconstrained, i.e. this can be any LTL-like formula. This is a newly introduced

logic although its definition follows a classical schema for branching-time temporal logics. Given a set of agents $Agt = \{a_1, \ldots, a_k\}$ and $r \geq 1$, we write RB±ATL$^*(Agt, r)$ to denote the resource-bounded logic with k agents and r resources whose models are resource-bounded concurrent game structures with the same parameters. Formulae of RB±ATL$^*(Agt, r)$ are defined according to the grammar below (we distinguish state formulae from path formulae as this can be done with CTL* or for RBTL*)

$$\begin{array}{rcl} \phi & ::= & p \mid \neg\phi \mid \phi \wedge \phi \mid \langle\langle A^{\boldsymbol{b}} \rangle\rangle \Phi \\ \Phi & ::= & \phi \mid \neg\Phi \mid (\Phi \wedge \Phi) \mid \bigcirc \Phi \mid (\Phi \boldsymbol{\mathcal{U}} \Phi) \mid \Box\Phi \end{array}$$

where $p \in$ PROP, $A \subseteq Agt$ and $\boldsymbol{b} \in (\mathbb{N} \cup \{\omega\})^r$. So, the set of state formulae ϕ for RB±ATL$^*(Agt, r)$ extends the set of formulae for RB±ATL(Agt, r). Strictly speaking, $\Box$ can be encoded with $\boldsymbol{\mathcal{U}}$ and $\neg$ but we keep it to emphasize that we are dealing with an extension of RB±ATL(Agt, r). We present the semantics for RB±ATL* in a style where the distinction between state formulae and path formulae is explicit. The two satisfaction relations $\models_{\mathrm{s}}$ and $\models_{\mathrm{p}}$ are defined as follows.

$\mathfrak{M}, s \models_{\mathrm{s}} p$	iff	$s \in L(p)$
$\mathfrak{M}, s \models_{\mathrm{s}} \langle\langle A^{\boldsymbol{b}} \rangle\rangle \Phi$	$\stackrel{\text{def}}{\Leftrightarrow}$	there is a $\boldsymbol{b}$-strategy F_A such that
		for all $\lambda = s_0 \xrightarrow{\mathrm{f}_0} s_1 \ldots \in \mathtt{out}(s, F_A)$, we have $\mathfrak{M}, \lambda \models_{\mathrm{p}} \Phi$
$\mathfrak{M}, \lambda \models_{\mathrm{p}} \phi$	iff	$\mathfrak{M}, \lambda(0) \models_{\mathrm{s}} \phi$ for state formulae ϕ
$\mathfrak{M}, \lambda \models_{\mathrm{p}} \neg\Phi$	iff	$\mathfrak{M}, \lambda \not\models_{\mathrm{p}} \Phi$
$\mathfrak{M}, \lambda \models_{\mathrm{p}} \Phi \wedge \Phi'$	iff	$\mathfrak{M}, \lambda \models_{\mathrm{p}} \Phi$ and $\mathfrak{M}, \lambda \models_{\mathrm{p}} \Phi'$
$\mathfrak{M}, \lambda \models_{\mathrm{p}} \bigcirc\Phi$	iff	$\mathfrak{M}, \lambda[1, +\infty) \models_{\mathrm{p}} \Phi$
$\mathfrak{M}, \lambda \models_{\mathrm{p}} \Box\Phi$	iff	$\mathfrak{M}, \lambda[i, +\infty) \models_{\mathrm{p}} \Phi$ for all $i < \lvert\lambda\rvert$
$\mathfrak{M}, \lambda \models_{\mathrm{p}} \Phi \boldsymbol{\mathcal{U}} \Psi$	iff	there is $i < \lvert\lambda\rvert$ such that $\mathfrak{M}, \lambda[i, +\infty) \models_{\mathrm{p}} \Psi$ and
		for every $j \in [0, i-1]$, we have $\mathfrak{M}, \lambda[j, +\infty) \models_{\mathrm{p}} \Phi$.

Again, all the maximal computations are infinite, the index i involved for clauses related to $\Box$ or $\boldsymbol{\mathcal{U}}$ can take any value in $\mathbb{N}$. The *model-checking problem for RB±ATL** is defined as follows:

Input: $k, r \geq 1$ (in unary), a state formula ϕ in RB±ATL$^*([1, k], r)$, a finite model $\mathfrak{M}$ and a state s,
Question: $\mathfrak{M}, s \models_{\mathrm{s}} \phi$?

Below, we show that the model-checking problem for RB±ATL* is decidable by reduction to the parity game problem for single-sided VASS [AMSS13, Corollary 2]. The latter problem will play a role similar to LTL model-checking to perform CTL* model-checking, see e.g. [Sch03]. Actually, [AMSS13, Theorem 4] allows us to synthetise resource bounds. First, let us define the exact problem we have in mind. The *parameterised version* of RB±ATL* denoted by ParRB±ATL* admits formulae as RB±ATL* except that the concrete values $\boldsymbol{b} \in (\mathbb{N} \cup \{\omega\})^r$ are replaced by tuples of variables/parameters within $\mathtt{VAR} = \{\mathtt{x}_1, \mathtt{x}_2, \ldots\}$. Here is a typical formula in ParRB±ATL*:

$$\langle\langle\{1\}^{(\mathtt{x}_1, \mathtt{x}_2)}\rangle\rangle \top \boldsymbol{\mathcal{U}} q_f \wedge \langle\langle\{2\}^{(\mathtt{x}_2, \mathtt{x}_3)}\rangle\rangle \top \boldsymbol{\mathcal{U}} q'_f.$$

Given a parameterised (state or path) formula ϕ with variables $\mathtt{x}, \ldots, \mathtt{x}_n$ and a map $\mathfrak{v} : \{\mathtt{x}_1, \ldots, \mathtt{x}_n\} \to (\mathbb{N} \cup \{\omega\})$, we write $\mathfrak{v}(\phi)$ to denote the formula in RB±ATL* obtained from ϕ by replacing each occurrence of a variable $\mathtt{x}$ by $\mathfrak{v}(\mathtt{x})$. The *parameterised model-checking problem for ParRB±ATL** is defined as follows:

Input: $k, r \geq 1$ (in unary), a parameterised state formula ϕ in ParRB±ATL$^*([1,k], r)$, a finite RB±ATL$^*([1,k], r)$ model $\mathfrak{M}$ and a state s,
Question: Compute the set of maps $\mathfrak{v}$ such that $\mathfrak{M}, s \models_s \mathfrak{v}(\phi)$.

Here, computing the set of maps means to be able to characterise the set of maps $\mathfrak{v}$ with $\mathfrak{M}, s \models_s \mathfrak{v}(\phi)$, by using a symbolic representation with nice computational properties. We need to be a bit more precise here. We will show that we only need Boolean combinations of atomic formulae of the form either $\mathtt{x} \geq k$ where $k \in \mathbb{N}$ or $\mathtt{x} = \omega$. With such contraints it is easy to check non-emptiness or to check the satisfaction of $\mathfrak{M}, s \models_s \mathfrak{v}(\phi)$ for a specific map $\mathfrak{v}$. To be able to synthetise such parameters we use a remarkable result about single-sided VASS: the Pareto frontier for any parity game on single-sided VASS is computable [AMSS13, Theorem 4].

Below, we consider AVASS with a finite set of fork rules included in $\bigcup_{\beta \geq 2} Q^\beta$, and the proofs are trees with nodes labelled by elements in $Q \times (\mathbb{N} \cup \{\omega\})^r$. Given an AVASS $\mathcal{A} = (Q, r, R_1, R_2)$, a *colouring* $\mathtt{col}$ is defined as a map $Q \to [0, p]$ for some $p \geq 0$. The *parity game problem* for AVASS (a.k.a. single-sided VASS) is defined as follows:

Input: An alternating VASS $\mathcal{A}$, a control state q_0, $\boldsymbol{b} \in (\mathbb{N} \cup \{\omega\})^r$ and $\mathtt{col} : Q \to [0, p]$.
Question: Is there is a proof whose root is equal to $(q_0, \boldsymbol{b})$, all the maximal branches are infinite and the maximal colour that appears infinitely often is even (the colour of each configuration is induced by $\mathtt{col}$) ?

Proposition 4. *[AMSS13, Corollary 2] The parity game problem for alternating VASS is decidable.*

In order to be precise, [AMSS13, Corollary 2] states the result for single-sided VASS that can be viewed as alternating VASS such that the set Q of control states can be partitioned into $Q = Q_1 \uplus Q_2$, unary rules start by states in Q_1, fork rules start by states in Q_2 and there is at most one fork rule starting by the same control state (necessarily, it belongs to Q_2). Existence of two disjoint sets Q_1 and Q_2 with alternation of unary rules and fork rules can be done as in the part (1) in the proof of Theorem 2. However, the colour of the new control states is equal to zero to avoid any new constraint. In order to guarantee unicity of fork rules starting by a given control state, it is sufficient to replace any unary rule $q \xrightarrow{\boldsymbol{u}} q'$ and fork rule $\mathfrak{r} = (q', q_1, q_2)$ by the unary rule $q \xrightarrow{\boldsymbol{u}} (q', \mathfrak{r})$ and the fork rule $((q', \mathfrak{r}), q_1, q_2)$. The colour of $(q', \mathfrak{r})$ is precisely the colour of q'. With such a transformation, decidability on single-sided VASS can be lifted to alternating VASS (this implies also a reduction for the computation of the Pareto frontier below).

It is not difficult to show that the state reachability and non-termination problems for AVASS can be understood as subproblems of the parity game problem and therefore their decidability follows also from [AMSS13]. However, in terms of computational complexity the situation is quite different. Whereas the exact complexity of the parity game problem is not yet known, the state reachability and non-termination problems for AVASS are shown 2EXPTIME-hard in [CS14], the state reachability problem

is shown in 2EXPTIME in [CS14] and the non-termination problem is proved in 2EXPTIME in [JLS15].

This decidability result has been strenghtened in [AMSS13] in the following way. Given $\mathcal{A}$, q_0 and $\mathtt{col} : Q \to [0,p]$, the set of tuples $\boldsymbol{b} \in (\mathbb{N} \cup \{\omega\})^r$ for which there is a proof such that the root is equal to $(q_0, \boldsymbol{b})$, all the maximal branches are infinite and for each infinite branch, the maximal colour that appears infinitely often is even, is upward closed and computable. This means that it can be represented effectively by a Boolean combination of atomic constraints of the form either $\mathtt{x}_i \geq k$ where $i \in [1,r]$ and $k \in \mathbb{N}$ or $\mathtt{x}_i = \omega$. Since the set is upward closed, by Dickson's Lemma, it has a finite set of minimal elements (with respect to the well-quasi-ordering $\preceq$ slightly extended to accomodate the addition of the value ω) that allows to define easily the symbolic representation in terms of atomic constraints of the form $\mathtt{x} \geq k$. The *Pareto frontier* of $\mathcal{A}$, q_0 and $\mathtt{col} : Q \to [0,p]$ is defined as the set of minimal elements.

Proposition 5. *[AMSS13, Theorem 4] The Pareto frontier for any parity game on single-sided VASS is computable.*

Before defining the reduction from the model-checking problem for RB±ATL* into the parity game problem, we need to introduce a few more definitions and in particular a notion of synchronisation that will be handy. Let $\mathfrak{M} = (Agt, S, Act, r, \mathtt{act}, \mathtt{cost}, \delta, L)$ be a resource-bounded concurrent game structure. Given the propositional variables $p_1, \ldots, p_n$, we write $\Sigma_n \stackrel{\text{def}}{=} \mathcal{P}(\{p_1, \ldots, p_n\})$ to denote the finite alphabet and $L_n(s') \stackrel{\text{def}}{=} \{p_i \mid i \in [1,n],\ s' \in L(p_i)\}$ for all $s' \in S$. So, $L_n(s') \in \Sigma_n$.

Let $\mathcal{A}_{\mathfrak{M},A,s} = (Q, r, R_1, R_2)$ be the AVASS defined from $\mathfrak{M}$, A and s (see Section 4.1), $\mathbb{A} = (Q', q'_0, \delta : Q' \times \Sigma_n \to Q', \mathtt{col} : Q' \to [0,p])$ be a deterministic parity automaton over the alphabet Σ_n. The principle of the *synchronised product* $\mathcal{A}_{\mathfrak{M},A,s} \otimes \mathbb{A}$ defined below is the following. Any (infinite) branch of a proof of $\mathcal{A}_{\mathfrak{M},A,s}$ contains control states of the form s, $(s', \mathfrak{f})$ or $(\mathfrak{g}, s')$ where s is a distinguished state of $\mathfrak{M}$, s' is any state, $\mathfrak{f} \in D_A(s')$ and $\mathfrak{g}$ is joint action in $D(s'')$ with $\delta(s'', \mathfrak{g}) = s'$. By contruction, $(s', \mathfrak{f})$ is preceded by a state of the form either $(\mathfrak{g}, s')$ or s' (if $s' = s$). So an infinite branch of the form $(s_0, \boldsymbol{u}_0)\ ((s_0, \mathfrak{f}_0), \boldsymbol{u}_1)\ ((\mathfrak{g}_1, s_1), \boldsymbol{u}_1)\ ((s_1, \mathfrak{f}_1), \boldsymbol{u}_2)\ ((\mathfrak{g}_2, s_2), \boldsymbol{u}_2) \cdots$ leads to the ω-word $L_n(s_0)\, L_n(s_1)\, L_n(s_2) \cdots$ that admits a unique run in $\mathbb{A}$ (because $\mathbb{A}$ is deterministic). Above, we slightly abuse notation since we identify a branch with its label. The control states of $\mathcal{A}_{\mathfrak{M},A,s} \otimes \mathbb{A}$ are pairs in $Q \times Q'$ and the second components are therefore control states in Q' as they appear for the run on $L_n(s_0)\, L_n(s_1)\, L_n(s_2) \cdots$.

Let us define the AVASS $\mathcal{A}_{\mathfrak{M},A,s} \otimes \mathbb{A} \stackrel{\text{def}}{=} (Q'', r, R'_1, R'_2)$ such that:

- $Q'' \stackrel{\text{def}}{=} Q \times Q'$.
- For each unary rule $s \xrightarrow{\boldsymbol{u}} (s, \mathfrak{f}) \in R_1$, we consider in R'_1 the unary rule $(s, q'_0) \xrightarrow{\boldsymbol{u}} ((s, \mathfrak{f}), q'_0)$.
- For each unary rule $(\mathfrak{g}, s') \xrightarrow{\boldsymbol{u}} (s', \mathfrak{f}) \in R_1$, and for each $q \in Q'$, we consider in R'_1 the unary rule $((\mathfrak{g}, s'), q) \xrightarrow{\boldsymbol{u}} ((s', \mathfrak{f}), q)$. So, firing a unary rule from $\mathcal{A}_{\mathfrak{M},A,s}$ does not change the second component.
- For each fork rule $((s', \mathfrak{f}), (\mathfrak{g}_1, s_1), \ldots, (\mathfrak{g}_\alpha, s_\alpha)) \in R_2$, and for each $q \in Q'$, we consider in R'_2 the fork rule

$$(((s', \mathfrak{f}), q), ((\mathfrak{g}_1, s_1), \delta(q, L_n(s'))), \ldots, ((\mathfrak{g}_\alpha, s_\alpha), \delta(q, L_n(s')))).$$

So, firing a fork rule from $\mathcal{A}_{\mathfrak{M},A,s}$ does change the second component in a unique way depending on q and on the letter $L_n(s')$. Again, there is a unique fork rule starting by the control state $((s',\mathfrak{f}),q)$.

Let us define the colouring $\mathtt{col}' : Q'' \to [0,p]$ such that for all $(q,q') \in Q''$, we have $\mathtt{col}'((q,q')) \stackrel{\text{def}}{=} \mathtt{col}(q')$. This is the most natural way to inherit colours from $\mathbb{A}$ to $\mathcal{A}_{\mathfrak{M},A,s} \otimes \mathbb{A}$.

Lemma 10. *Let $(s,\boldsymbol{b}) \in Q \times (\mathbb{N} \cup \{\omega\})^r$. There is an equivalence between the statements below:*

(I) *$\mathcal{A}_{\mathfrak{M},A,s}$ has a proof whose root is equal to $(s,\boldsymbol{b})$, all the maximal branches are infinite and the L_n-projection of each infinite branch belongs to $\mathrm{L}(\mathbb{A})$.*

(II) *$\mathcal{A}_{\mathfrak{M},A,s} \otimes \mathbb{A}$ has a proof whose root is equal to $((s,q'_0),\boldsymbol{b})$, all the maximal branches are infinite and for all infinite branches, the maximal colour that appears infinitely often is even (based on the colouring function $\mathtt{col}'$) ?*

Given an infinite branch

$$s_0 \xrightarrow{\boldsymbol{u}_0} (s_0,\mathfrak{f}_0) \to (\mathfrak{g}^1_{k_1}, s^1_{k_1}) \xrightarrow{\boldsymbol{u}_1} (s^1_{k_1},\mathfrak{f}_1) \to (\mathfrak{g}^2_{k_2}, s^2_{k_2}) \xrightarrow{\boldsymbol{u}_2} (s^2_{k_2},\mathfrak{f}_2) \to (\mathfrak{g}^3_{k_3}, s^3_{k_3}) \cdots$$

in a proof of $\mathcal{A}_{\mathfrak{M},A,s}$, its *$L_n$-projection* is simply defined as the ω-word in Σ_n^ω below:

$$L_n(s_0)\, L_n(s^1_{k_1})\, L_n(s^2_{k_2})\, L_n(s^3_{k_3}) \cdots$$

Proof. (I) $\to$ (II) Let $\hat{\mathcal{D}} : \mathfrak{T} \to Q \times \mathbb{N}^r$ be a proof of $\mathcal{A}_{\mathfrak{M},A,s}$ such that $\hat{\mathcal{D}}(\varepsilon) = (s,\boldsymbol{b})$, all the maximal branches are infinite and their L_n-projections belong to $\mathrm{L}(\mathbb{A})$. This means that for any label

$$s_0 \xrightarrow{\boldsymbol{u}_0} (s_0,\mathfrak{f}_0) \to (\mathfrak{g}^1_{k_1}, s^1_{k_1}) \xrightarrow{\boldsymbol{u}_1} (s^1_{k_1},\mathfrak{f}_1) \to (\mathfrak{g}^2_{k_2}, s^2_{k_2}) \xrightarrow{\boldsymbol{u}_2} (s^2_{k_2},\mathfrak{f}_2) \to \cdots$$

of an infinite branch, we have $L_n(s_0)\, L_n(s^1_{k_1})\, L_n(s^2_{k_2})\, L_n(s^3_{k_3}) \cdots \in \mathrm{L}(\mathbb{A})$.

Let $\hat{\mathcal{D}}' : \mathfrak{T} \to (Q \times Q') \times \mathbb{N}^r$ be the map defined below that will turn out to be a proof of $\mathcal{A}_{\mathfrak{M},A,s} \otimes \mathbb{A}$ built over the same infinite tree $\mathfrak{T}$. Let $i_1 i_2 i_3 \cdots$ be an infinite branch with the above-mentioned label such that $L_n(s_0)\, L_n(s^1_{k_1})\, L_n(s^2_{k_2})\, L_n(s^3_{k_3}) \cdots \in \mathrm{L}(\mathbb{A})$. Since $\mathbb{A}$ is deterministic, there is a unique (accepting) run $q'_0 \xrightarrow{L_n(s_0)} q'_1 \xrightarrow{L_n(s^1_{k_1})} q'_2 \cdots$, whence the maximal colour that appears infinitely often is even. For any finite prefix $\mathfrak{w} \sqsubset i_1 i_2 i_3 \cdots$ of length N, we have

$$\hat{\mathcal{D}}'(\mathfrak{w}) \stackrel{\text{def}}{=} ((q, q'_{\lfloor \frac{N}{2} \rfloor}), \boldsymbol{v}) \text{ where } \hat{\mathcal{D}}(\mathfrak{w}) = (q,\boldsymbol{v}).$$

Since $\mathbb{A}$ is deterministic, this allows to define uniquely the map $\hat{\mathcal{D}}'$. Indeed, classically, if $\mathbb{A}$ were not deterministic, we cannot guarantee that two infinite words in $\mathrm{L}(\mathbb{A})$ sharing a common non-empty prefix have the same subrun for that prefix. As $\mathcal{A}_{\mathfrak{M},A,s} \otimes \mathbb{A}$ is also designed as the synchronised product between $\mathcal{A}_{\mathfrak{M},A,s}$ and $\mathbb{A}$, one can check that

$\hat{\mathcal{D}}'$ is indeed a proof of $\mathcal{A}_{\mathfrak{M},A,s} \otimes \mathbb{A}$ such that $\hat{\mathcal{D}}'(\varepsilon) = ((s, q'_0), \boldsymbol{b})$ and all the maximal branches are infinite. Consider below the label of any infinite branch:

$$(s_0, q'_0) \xrightarrow{\boldsymbol{u}_0} ((s_0, \mathfrak{f}_0), q'_0) \to ((\mathfrak{g}^1_{k_1}, s^1_{k_1}), q'_1) \xrightarrow{\boldsymbol{u}_1} ((s^1_{k_1}, \mathfrak{f}_1), q'_1) \to ((\mathfrak{g}^2_{k_2}, s^2_{k_2}), q'_2) \xrightarrow{\boldsymbol{u}_2} \cdots$$

Since $q'_0 \xrightarrow{L_n(s_0)} q'_1 \xrightarrow{L_n(s^1_{k_1})} q'_2 \cdots$ is an accepting run, the maximal colour that appears infinitely often on the branch is even.

(II) $\to$ (I) Let $\hat{\mathcal{D}} : \mathfrak{T} \to (Q \times Q') \times \mathbb{N}^r$ be a proof of $\mathcal{A}_{\mathfrak{M},A,s} \otimes \mathbb{A}$ whose root is $((s, q'_0), \boldsymbol{b})$, all the maximal branches are infinite and the maximal colour that appears infinitely often is even. Let $\hat{\mathcal{D}}' : \mathfrak{T} \to Q \times \mathbb{N}^r$ be the map defined below that will turn out to be a proof of $\mathcal{A}_{\mathfrak{M},A,s}$ and that can be viewed as $\hat{\mathcal{D}}$ where the component in Q' is omitted. For all $\mathfrak{w} \in \mathfrak{T}$, $\hat{\mathcal{D}}'(\mathfrak{w}) \stackrel{\text{def}}{=} (q, \boldsymbol{v})$ where $\hat{\mathcal{D}}(\mathfrak{w}) = ((q, q'), \boldsymbol{v})$. We have $\hat{\mathcal{D}}(\varepsilon) = (s, \boldsymbol{b})$ and all the maximal branches are infinite. Consider below the label of an infinite branch $i_1 i_2 i_3 \cdots$:

$$s_0 \xrightarrow{\boldsymbol{u}_0} (s_0, \mathfrak{f}_0) \to (\mathfrak{g}^1_{k_1}, s^1_{k_1}) \xrightarrow{\boldsymbol{u}_1} (s^1_{k_1}, \mathfrak{f}_1) \to (\mathfrak{g}^2_{k_2}, s^2_{k_2}) \xrightarrow{\boldsymbol{u}_2} (s^2_{k_2}, \mathfrak{f}_2) \to \cdots$$

Since $\hat{\mathcal{D}}'$ is obtained from $\hat{\mathcal{D}}$ by projection, the label of $i_1 i_2 i_3 \cdots$ in $\hat{\mathcal{D}}$ is of the form

$$(s_0, q'_0) \xrightarrow{\boldsymbol{u}_0} ((s_0, \mathfrak{f}_0), q'_0) \to ((\mathfrak{g}^1_{k_1}, s^1_{k_1}), q'_1) \xrightarrow{\boldsymbol{u}_1} ((s^1_{k_1}, \mathfrak{f}_1), q'_1) \to ((\mathfrak{g}^2_{k_2}, s^2_{k_2}), q'_2) \xrightarrow{\boldsymbol{u}_2} \cdots$$

By assumption on $\hat{\mathcal{D}}$, the maximal colour that appears infinitely often is even and therefore $q'_0 \xrightarrow{L_n(s_0)} q'_1 \xrightarrow{L_n(s^1_{k_1})} q'_2 \cdots$ is an accepting run of $\mathbb{A}$, whence

$$L_n(s_0)\, L_n(s^1_{k_1})\, L_n(s^2_{k_2})\, L_n(s^3_{k_3}) \cdots \in \mathrm{L}(\mathbb{A}).$$

Theorem 5. *The model-checking problem for RB±ATL* is decidable.*

Proof. The model-checking problem for RB±ATL* is solved by using as subroutine the algorithm for solving the parity game problem for alternating VASS with a simple renaming technique. The algorithm uses a dynamic programming approach that first computes on which states the subformulae hold before dealing with larger formulae. This is indeed a labelling algorithm whose principles have been already applied many times. However, again, there is a caveat: when dealing with subformulae of the form $\langle\langle A^{\boldsymbol{b}} \rangle\rangle\, \Phi$ where Φ is a path formula without any strategy modality, we are entitled to use the algorithm to solve the parity game problem for alternating VASS. However, in order to systematically consider such subformulae $\langle\langle A^{\boldsymbol{b}} \rangle\rangle\, \Phi$ when the outermost connective is a strategy modality, we need to perform renamings on-the-fly.

Let ϕ be a formula built over the propositional variables $\{p_1, \ldots, p_n\}$ and $\langle\langle A^{\boldsymbol{b}} \rangle\rangle\, \Phi$ be one of its subformulae such that no strategy modality occurs in Φ. Without any loss of generality, we can assume that there is an injective map $\mathtt{nom} : S \to [1, n]$ such that for every $s \in S$, $L(p_{\mathtt{nom}(s)}) = \{s\}$. So, each propositional variable $p_{\mathtt{nom}(s)}$ holds true on a single state, namely on s. So, even though we assume that ϕ is built over $\{p_1, \ldots, p_n\}$, we do not require that all the propositional variables in $\{p_1, \ldots, p_n\}$ necessarily occur in ϕ (some of the propositional variables are only used to name states). Given a finite

game structure $\mathfrak{M}$, it is always possible to enrich it so that each state can be named by a dedicated propositional variable (also called a nominal in hybrid logics). This can be done in linear time.

Since Φ is an LTL formula built over $\{p_1, \ldots, p_n\}$, there is a Büchi automaton $\mathbb{A}$ over the alphabet Σ_n such that $\mathrm{L}(\mathbb{A})$ is equal to the set of models of Φ (over the set of propositional variables $\{p_1, \ldots, p_n\}$), see e.g. [VW94]. Say that $\mathbb{A}$ has N states and N is in $O(2^{|\Phi|})$. Since Büchi automata and deterministic parity automata both recognize the set of ω-regular languages, there is deterministic parity automaton $\mathbb{B}$ with initial location q_0', $O(N!^2)$ states and $2N$ priorities such that $\mathrm{L}(\mathbb{A}) = \mathrm{L}(\mathbb{B})$ [Sch09]. The automaton $\mathbb{B}$ can be effectively computed from $\mathbb{A}$. A construction similar to the one for $\mathbb{B}$ has been recently also used for model-checking pushdown multi-agent systems [CSW16].

Let $X \subseteq S$ be the set of states s such that $\mathcal{A}_{\mathfrak{M},A,s} \otimes \mathbb{B}$ has a proof whose root is equal to $((s, q_0'), \boldsymbol{b})$, all the maximal branches are infinite and the maximal colour that appears infinitely often is even. We update the formula ϕ by replacing every occurrence of $\langle\langle A^b \rangle\rangle \Phi$ by $\psi = \bigvee_{s' \in X} p_{\mathtt{nom}(s')}$. The set X can be computed thanks to Proposition 4 and this is a correct step thanks to Lemma 10 and Lemma 4. Indeed, for all $s' \in S$, we have $\mathfrak{M}, s' \models_{\mathrm{s}} \langle\langle A^b \rangle\rangle \Phi$ iff $\mathfrak{M}, s' \models_{\mathrm{s}} \bigvee_{s' \in X} p_{\mathtt{nom}(s')}$ and therefore, for all $s' \in S$, we have $\mathfrak{M}, s' \models_{\mathrm{s}} \phi$ iff $\mathfrak{M}, s' \models_{\mathrm{s}} \phi[\psi / \langle\langle A^b \rangle\rangle \Phi]$ where $\phi[\psi / \langle\langle A^b \rangle\rangle \Phi]$ is obtained from ϕ by substituting every occurrence of $\langle\langle A^b \rangle\rangle \Phi$ by ψ. We update ϕ until there is no more strategy modality and therefore eventually ϕ is a Boolean combination of propositional variables, which is then easy to evaluate on a given state.

The proof of Theorem 5 uses a synchronised product between an alternating VASS and a deterministic parity automaton recognizing ω-words. This is reminiscent of the proof of [AHK02, Theorem 5.6] about the 2EXPTIME upper bound for ATL* model-checking problem. Rabin tree automata from the proof of [AHK02, Theorem 5.6] are replaced by deterministic parity automata for encoding the LTL formulae and by alternating VASS (with counters) as outcome of the synchronisation.

So, decidability is quite a strong result by combining the previous developments and results from [AMSS13]. This can be improved by synthetising values for parameters by adequately taking advantage of the effective computation of the Pareto frontiers from [AMSS13]. This is the subject of the next section.

5.3 Symbolic representations for sets of resource values

Let $\mathfrak{M} = (Agt, S, Act, r, \mathtt{act}, \mathtt{cost}, \delta, L)$ be a resource-bounded concurrent game structure and ϕ be a ParRB±ATL* formula such that the resource variables are among $\mathtt{x}_1, \ldots, \mathtt{x}_m$ and the propositional variables are among $p_1, \ldots, p_n$.

We write $\varphi_1 = \langle\langle A_1^{\mathtt{t}_1} \rangle\rangle \phi_1$, …, $\varphi_\alpha = \langle\langle A_\alpha^{\mathtt{t}_\alpha} \rangle\rangle \phi_\alpha$ to denote the subformulae of ϕ whose outermost connective is a strategy modality and the subformulae are arranged by increasing size. So, ϕ_1 does not contain any strategy modality and it can be viewed as an LTL formula built over $\{p_1, \ldots, p_n\}$. Each $\mathtt{t}_i$ is a tuple of r variables (say $\mathtt{t}_i = (\mathtt{t}_i^1, \ldots, \mathtt{t}_i^r)$) and not only a variable can occur strictly more than once in $\mathtt{t}_i$ but two distinct tuples $\mathtt{t}_i$ and $\mathtt{t}_j$ can share variables, which provides a great flexibility in the logical formalism. Below, for each state $s \in S$ and for all $i \in [1, \alpha]$, we build a formula

$\hat{\varphi}_i^s$ over $\mathtt{x}_1, \ldots, \mathtt{x}_m$ following the grammar below:

$$\psi ::= \mathtt{x} \geq k \mid \mathtt{x} = \omega \mid \neg\psi \mid \psi \vee \psi \mid \psi \wedge \psi,$$

where $\mathtt{x} \in \mathtt{VAR}$ and $k \in \mathbb{N}$. Such formulae are interpreted over valuations $\mathfrak{v} : \mathtt{VAR} \to \mathbb{N} \cup \{\omega\}$ with the expected semantics based on the satisfaction relation $\mathfrak{v} \models \psi$. The key property is the following one: for all $\mathfrak{v}$, we have $\mathfrak{v} \models \hat{\varphi}_i^s$ iff $\mathfrak{M}, s \models \mathfrak{v}(\langle\langle A_i^{\mathtt{t}_i} \rangle\rangle \phi_i)$. Note that ϕ_i may contain variables.

Before explaining how to construct the formulae $\hat{\varphi}_i^s$'s, let us explain how to construct from such formulae a constrained formula ψ_s such that for all $\mathfrak{v}$, we have $\mathfrak{v} \models \psi_s$ iff $\mathfrak{M}, s \models \mathfrak{v}(\phi)$. Let $\varphi_{i_1}, \ldots, \varphi_{i_z}$ be the maximal subformulae of ϕ such that its outermost connective is a strategy modality. Given a propositional valuation $\mathfrak{h} : \{\varphi_{i_1}, \ldots, \varphi_{i_s}\} \to \{\bot, \top\}$, we define $\mathfrak{M}, s \models \mathfrak{h}(\phi)$ iff $\mathfrak{h}(\phi)$ obtained from ϕ by replacing simultaneously each φ_{i_j} by $\mathfrak{h}(\varphi_{i_j})$ is evaluated to true in s. The formula ψ_s is defined as follows:

$$\bigvee_{\mathfrak{h},\ \mathfrak{M},s \models \mathfrak{h}(\phi)} \Big(\bigwedge_{j, \mathfrak{h}(\varphi_{i_j}) = \top} \hat{\varphi}_{i_j}^s \Big) \wedge \Big(\bigwedge_{j, \mathfrak{h}(\varphi_{i_j}) = \bot} \neg\hat{\varphi}_{i_j}^s \Big).$$

Let us explain now how to build $\hat{\varphi}_i^s$. Let $\varphi_{j_1}, \ldots, \varphi_{j_\beta}$ be the maximal subformulae of ϕ_i such that its outermost connective is a strategy modality. By assumption about the increasing size, we have $\{j_1, \ldots, j_\beta\} \subseteq [1, i-1]$ and possibly, there are none such subformulae in the case ϕ_i has no strategy modality. Given $S_1, \ldots, S_\beta \subseteq S$ and $I \subseteq [1, r]$, let $\psi^s_{I, S_1, \ldots, S_\beta}$ be the formula encoding the Pareto frontier of $\mathcal{A}_{\mathfrak{M}, A_i, s} \otimes \mathbb{A}$ with initial state (s, q_0'), ω-components in I exactly and associated colouring such that $\mathbb{A}$ is the deterministic parity automaton of L. This latter language is actually an ω-regular language over the alphabet $\mathcal{P}(\{p_1, \ldots, p_n\})$ defined by the LTL formula ϕ_i' that is obtained from ϕ_i by replacing every occurrence of φ_j by

$$\Big(\bigvee_{s \in S_j} p_{\mathrm{nom}(s)}\Big) \wedge \Big(\bigwedge_{s \notin S_j} \neg p_{\mathrm{nom}(s)}\Big).$$

The formula $\hat{\varphi}_i^s$ is then defined as the following disjunction:

$$\bigvee_{S_1, \ldots, S_\beta \subseteq S, I \subseteq [1,r]} \psi^s_{I, S_1, \ldots, S_\beta} \wedge \Big(\bigwedge_{\gamma \in [1,\beta]} \Big(\bigwedge_{s' \in S_\gamma} \hat{\varphi}_{j_\gamma}^{s'} \Big) \wedge \Big(\bigwedge_{s' \notin S_\gamma} \neg\hat{\varphi}_{j_\gamma}^{s'} \Big) \Big).$$

The proof of forthcoming Lemma 11 provides the justification for such a construction.

Note that since $\{j_1, \ldots, j_\beta\} \subseteq [1, i-1]$, each formula $\hat{\varphi}_{j_\gamma}^{s'}$ is already defined and therefore the formula $\hat{\varphi}_i^s$ can be safely built.

Lemma 11. *For all $i \in [1, \alpha]$, for all concretisations $\mathfrak{v}$, we have $\mathfrak{v} \models \hat{\varphi}_i^s$ iff $\mathfrak{M}, s \models \mathfrak{v}(\langle\langle A_i^{\mathtt{t}_i} \rangle\rangle \phi_i)$.*

Proof. The proof is by induction on i.
Base case: there is no strategy modality in ϕ_i (this includes the case $i = 1$).
Recall that $\varphi_i = \langle\langle A_i^{\mathtt{t}_i} \rangle\rangle \phi_i$. Since there is no strategy modality in ϕ_i, the formula ϕ_i is

simply an LTL formula built over propositional variables in $\{p_1, \ldots, p_n\}$ and let $\mathbb{A}$ be a deterministic parity automaton such that the models of ϕ_i on Σ_n are precisely $\mathrm{L}(\mathbb{A})$. It is known that $\mathbb{A}$ can be effectively computed from ϕ_i.

Given $I \subseteq [1, r]$, we can construct a constrained formula ψ_I^s characterizing the Pareto frontier of $\mathcal{A}_{\mathfrak{M},A,s} \otimes \mathbb{A}$ with initial state (s, q_0') and the values at the positions in I are equal to ω at the root (see [AMSS13, Theorem 4]). Consequently ψ_I^s is equivalent to $\psi_I^s \wedge (\bigwedge_{j \in I} \mathtt{t}_i^j = \omega) \wedge (\bigwedge_{j \notin I} \mathtt{t}_i^j \neq \omega)$ where $\mathtt{t}_i = (\mathtt{t}_i^1, \ldots, \mathtt{t}_i^r)$. This means that for all $\boldsymbol{b} \in (\mathbb{N} \cup \{\omega\})^r$ such that (C_I) for all $j \in [1, r]$, $\boldsymbol{b}(j) = \omega$ iff $j \in I$, we have that $\boldsymbol{b} \models \psi_I^s$ (meaning $\mathfrak{v} \models \psi_I^s$ with $\mathfrak{v}(\mathtt{t}_i^j) \stackrel{\text{def}}{=} \boldsymbol{b}(j)$ for all $j \in [1, r]$) iff there is a proof of $\mathcal{A}_{\mathfrak{M},A,s} \otimes \mathbb{A}$ whose root is $((s, q_0'), \boldsymbol{b})$, all the maximal branches are infinite and the maximal colour that appears infinitely often is even.

By Lemma 10, for all $\boldsymbol{b} \in (\mathbb{N} \cup \{\omega\})^r$ such that (C_I), we have $\boldsymbol{b} \models \psi_I^s$ iff $\mathcal{A}_{\mathfrak{M},A_i,s}$ has a proof whose root is $(s, \boldsymbol{b})$, all the maximal branches are infinite and the L_n-projection of each infinite branch is in $\mathrm{L}(\mathbb{A})$. By Lemma 4, for all $\boldsymbol{b} \in (\mathbb{N} \cup \{\omega\})^r$ such that (C_I), we have $\boldsymbol{b} \models \psi_I^s$ iff there is a $\boldsymbol{b}$-consistent strategy F_{A_i} w.r.t. s in $\mathfrak{M}$ such that the set of computations $\mathtt{out}(s, F_{A_i})$ is included in $\mathrm{L}(\mathbb{A})$. So, for all $\boldsymbol{b} \in (\mathbb{N} \cup \{\omega\})^r$ such that (C_I), we have $\boldsymbol{b} \models \psi_I^s$ iff $\mathfrak{M}, s \models \langle\langle A_i^{\mathtt{t}_i} \rangle\rangle \phi_i$. Consequently, for all concretisations $\mathfrak{v}$ such that for all $j \in [1, r]$ $\mathfrak{v}(\mathtt{t}_i^j) = \omega$ iff $j \in I$, we have $\mathfrak{v} \models \psi_I^s$ iff $\mathfrak{M}, s \models \mathfrak{v}(\langle\langle A_i^{\mathtt{t}_i} \rangle\rangle \phi_i)$. The formula $\hat{\varphi}_i^s$ is defined as a generalised disjunction parameterised by all the possible values for I, i.e. $\hat{\varphi}_i^s \stackrel{\text{def}}{=} \bigvee_{I \subseteq [1,r]} \psi_I^s$, and it is easy to check that for all $\mathfrak{v}$, $\mathfrak{v} \models \hat{\varphi}_i^s$ is equivalent to $\mathfrak{M}, s \models \mathfrak{v}(\langle\langle A_i^{\mathtt{t}_i} \rangle\rangle \phi_i)$.

Induction step. $\varphi_i = \langle\langle A_i^{\mathtt{t}_i} \rangle\rangle \phi_i$ with $\varphi_{j_1}, \ldots, \varphi_{j_\beta}$ the maximal subformulae of ϕ_i ($\beta \geq 1$) such that its outermost connective is a strategy modality and for all $\gamma \in [1, \beta]$, for all $s' \in S$, for all $\mathfrak{v}$, we have $\mathfrak{v} \models \hat{\varphi}_{j_\gamma}^{s'}$ iff $\mathfrak{M}, s' \models \mathfrak{v}(\langle\langle A_{j_\gamma}^{\mathtt{t}_{j_\gamma}} \rangle\rangle \phi_{j_\gamma})$.

The proof for the induction step is quite similar to the proof for the base case except that we need to justify that the renaming mechanism we use is correct. First, let us state a few basic properties that shall be used below.

Given a state formula ϕ in RB±ATL*, we write $\mathfrak{M} \models \phi$ whenever for all $s' \in S$, we have $\mathfrak{M}, s' \models \phi$, i.e. ϕ is valid in $\mathfrak{M}$.

(P1) Let ϕ, ψ, ψ' be state formulae in RB±ATL* such that ψ occurs in ϕ and $\mathfrak{M} \models \psi \Leftrightarrow \psi'$. Then $\mathfrak{M} \models \phi \Leftrightarrow \phi[\psi'/\psi]$ where $\phi[\psi'/\psi]$ is defined from ϕ by replacing every occurrence of ψ by ψ'.

(P2) Let $\gamma \in [1, \beta]$ and $S' \subseteq S$. For all $\mathfrak{v}$, the statements below are equivalent:

- **(P2.1)** $\mathfrak{M} \models [(\bigvee_{s \in S'} p_{\mathtt{nom}(s)}) \wedge (\bigwedge_{s \notin S'} \neg p_{\mathtt{nom}(s)})] \Leftrightarrow \mathfrak{v}(\varphi_{j_\gamma})$.
- **(P2.2)** For all $s' \in S$, $\mathfrak{v} \models \hat{\varphi}_{j_\gamma}^{s'}$ iff $s' \in S'$.

The proof of (P1) is quite standard but it is worth noticing that the formulae ϕ, ψ and ψ' need to be state formulae. The proof of (P2) uses straightforwardly the induction hypothesis.

Let $S_1, \ldots, S_\beta \subseteq S$ and $I \subseteq [1, r]$. We write ϕ'_i to denote the formula obtained from ϕ_i replacing every occurrence of the propositional formula φ_{j_γ} by

$$\varphi^\star_{j_\gamma} \stackrel{\text{def}}{=} (\bigvee_{s' \in S_\gamma} p_{\text{nom}(s')}) \wedge (\bigwedge_{s' \notin S_\gamma} \neg p_{\text{nom}(s')}).$$

So, even though this is not made explicit in the notation, ϕ'_i depends on $S_1, \ldots, S_\beta$. Like the base case, the formula ϕ'_i is an LTL formula built over $\{p_1, \ldots, p_n\}$ and one can compute a deterministic parity automaton $\mathbb{A}$ such that the models of ϕ'_i on Σ_n are precisely $\mathrm{L}(\mathbb{A})$. We can also construct a constrained formula $\psi^s_{I,S_1,\ldots,S_\beta}$ characterising the Pareto frontier of $\mathcal{A}_{\mathfrak{M},A_i,s} \otimes \mathbb{A}$ with initial state (s, q'_0) and the values at the positions in I are equal to ω at the root, see [AMSS13, Theorem 4].

Following a reasoning analogous to the one from the base case (basically replace ϕ_i by ϕ'_i), we can conclude that for all concretisations $\mathfrak{v}$ such that for all $j \in [1, r]$ $\mathfrak{v}(\mathtt{t}_i^j) = \omega$ iff $j \in I$, we have $\mathfrak{v} \models \psi^s_{I,S_1,\ldots,S_\beta}$ iff $\mathfrak{M}, s \models \mathfrak{v}(\langle\langle A_i^{\mathtt{t}_i}\rangle\rangle \phi'_i)$ (†). However, what we really need is to consider $\langle\langle A_i^{\mathtt{t}_i}\rangle\rangle \phi_i$ (instead of $\langle\langle A_i^{\mathtt{t}_i}\rangle\rangle \phi'_i$) and this is the place where the induction hypothesis is again invoked.

First, let us introduce an auxiliary notion. A concretisation $\mathfrak{v}$ is said to *compatible with* $I, S_1, \ldots, S_\beta$ iff the conditions below hold:

1. $I = \{\gamma \in [1, r] \mid \mathfrak{v}(\mathtt{t}_i^\gamma) = \omega\}$.
2. For all $\gamma \in [1, \beta]$, $S_\gamma = \{s' \in S \mid \mathfrak{M}, s' \models \mathfrak{v}(\varphi_{j_\gamma})\}$.

Assuming that $\mathfrak{v}$ is compatible with $I, S_1, \ldots, S_\beta$, we can conclude the following facts:

- By induction hypothesis, for all $\gamma \in [1, \beta]$, $S_\gamma = \{s' \in S \mid \mathfrak{v} \models \hat{\varphi}^{s'}_{j_\gamma}\}$.
- By (P2), for all $\gamma \in [1, \beta]$, $\mathfrak{M} \models [(\bigvee_{s' \in S_\gamma} p_{\text{nom}(s')}) \wedge (\bigwedge_{s' \notin S_\gamma} \neg p_{\text{nom}(s')})] \Leftrightarrow \mathfrak{v}(\varphi_{j_\gamma})$.
- By (P1) and the newly established property (†),

$$\mathfrak{v} \models \psi^s_{I,S_1,\ldots,S_\beta} \text{ iff } \mathfrak{M}, s \models \mathfrak{v}(\langle\langle A_i^{\mathtt{t}_i}\rangle\rangle \phi'_i)[\varphi_{j_1}/\varphi^\star_{j_1}, \ldots, \varphi_{j_\beta}/\varphi^\star_{j_\beta}].$$

- Since $\mathfrak{v}(\phi'_i)[\varphi_{j_1}/\varphi^\star_{j_1}, \ldots, \varphi_{j_\beta}/\varphi^\star_{j_\beta}] = \mathfrak{v}(\phi_i)$, $\mathfrak{v} \models \psi^s_{I,S_1,\ldots,S_\beta}$ iff $\mathfrak{M}, s \models \mathfrak{v}(\langle\langle A_i^{\mathtt{t}_i}\rangle\rangle \phi_i)$.
- By using the compatibility of $\mathfrak{v}$, we get that $\mathfrak{v} \models \psi^s_{I,S_1,\ldots,S_\beta} \wedge (\bigwedge_{\gamma \in [1,\beta]} (\bigwedge_{s' \in S_\gamma} \hat{\varphi}^{s'}_{j_\gamma}) \wedge (\bigwedge_{s' \notin S_\gamma} \neg \hat{\varphi}^{s'}_{j_\gamma}))$ iff $\mathfrak{M}, s \models \mathfrak{v}(\langle\langle A_i^{\mathtt{t}_i}\rangle\rangle \phi_i)$.

The formula $\hat{\varphi}^s_i$ is defined as a generalised disjunction parameterised by all the possible values for $I, S_1, \ldots, S_\beta$, and it is easy to check that for all $\mathfrak{v}$, we have $\mathfrak{v} \models \hat{\varphi}^s_i$ equivalent to $\mathfrak{M}, s \models \mathfrak{v}(\langle\langle A_i^{\mathtt{t}_i}\rangle\rangle \phi_i)$.

Theorem 6. *The parameterised model-checking problem for ParRB±ATL* can be solved.*

Proof. Let ϕ be a state formula in $\mathrm{ParRB}{\pm}\mathrm{ATL}^*([1, k], r)$, $\mathfrak{M}$ be a resource-bounded concurrent game structure and $s \in S$. We write $\varphi_1 = \langle\langle A_1^{\mathtt{t}_1}\rangle\rangle \phi_1, \ldots, \varphi_z = \langle\langle A_z^{\mathtt{t}_z}\rangle\rangle \phi_z$ to denote the maximal subformulae of ϕ such that the outermost connective is a strategy modality. By Lemma 11, for all $j \in [1, z]$, for all valuations $\mathfrak{v} : \mathtt{VAR} \to \mathbb{N} \cup \{\omega\}$, we

have $\mathfrak{v} \models \hat{\varphi}_j^s$ iff $\mathfrak{M}, s \models \mathfrak{v}(\langle\langle A_j^{\mathtt{t}_j} \rangle\rangle \phi_j)$ where $\hat{\varphi}_j^s$ is a constrained formula that can be effectively built from ϕ, $\mathfrak{M}$ and s. Let ψ_s be the formula defined below:

$$\bigvee_{\mathfrak{h}:\{\varphi_1,\ldots,\varphi_z\}\to\{\bot,\top\},\ \mathfrak{M},s\models\mathfrak{h}(\phi)} \Big(\bigwedge_{j,\ \mathfrak{h}(\varphi_j)=\top} \hat{\varphi}_j^s \Big) \wedge \Big(\bigwedge_{j,\ \mathfrak{h}(\varphi_j)=\bot} \neg\hat{\varphi}_j^s \Big).$$

Each expression $\hat{\varphi}_j^s$ and $\neg\hat{\varphi}_j^s$ is a constrained formula, $\mathfrak{M}, s \models \mathfrak{h}(\phi)$ can be decided for any valuation $\mathfrak{h}$. Consequently, ψ_s defined above is a constrained formula that can be effectively computed.

Let $\mathfrak{v}$ be a concretisation such that $\mathfrak{M}, s \models \mathfrak{v}(\phi)$. There exists $\mathfrak{h}_0 : \{\varphi_1, \ldots, \varphi_z\} \to \{\bot, \top\}$ such that $\mathfrak{M}, s \models \mathfrak{h}_0(\phi)$ and for all $j \in [1, z]$, we have $\mathfrak{h}_0(\varphi_j) = \top$ iff $\mathfrak{M}, s \models \mathfrak{v}(\varphi_j)$. By Lemma 11, for all $j \in [1, z]$, we have $\mathfrak{h}_0(\varphi_j) = \top$ iff $\mathfrak{v} \models \hat{\varphi}_j^s$. So,

$$\mathfrak{v} \models \Big(\bigwedge_{j,\ \mathfrak{h}_0(\varphi_j)=\top} \hat{\varphi}_j^s \Big) \wedge \Big(\bigwedge_{j,\ \mathfrak{h}_0(\varphi_j)=\bot} \neg\hat{\varphi}_j^s \Big),$$

which entails $\mathfrak{v} \models \psi_s$.

Conversely, assuming that $\mathfrak{v} \models \psi_s$ for some concretisation $\mathfrak{v}$, there is a map $\mathfrak{h}$ such that $\mathfrak{v} \models (\bigwedge_{j,\ \mathfrak{h}(\varphi_j)=\top} \hat{\varphi}_j^s) \wedge (\bigwedge_{j,\ \mathfrak{h}(\varphi_j)=\bot} \neg\hat{\varphi}_j^s)$ and $\mathfrak{M}, s \models \mathfrak{h}(\phi)$. Again by Lemma 11, and by Boolean reasoning we obtain $\mathfrak{M}, s \models \mathfrak{v}(\phi)$. Consequently, the formula ψ_s can be computed and it is a symbolic representation for all the maps $\mathfrak{v}$ such that $\mathfrak{M}, s \models \mathfrak{v}(\phi)$.

6 Concluding Remarks

In this paper, we have related model-checking problems for resource-bounded logics and decision problems for alternating VASS, such as state reachability, non-termination and more generally parity game problems. Even though such relationships should not come as a complete surprise, we have been able to obtain new complexity and decidability results. Below, we recall some of them.

1. The model-checking problem for RB±ATL introduced in [ALNR14,ALNR15] is 2EXPTIME-complete. No complexity upper bound was known so far.
2. We have introduced the logic RB±ATL* that extends RB±ATL (as ATL* extends ATL), and we have shown that the model-checking problem is decidable. For the parameterised version ParRB±ATL*, given $\mathfrak{M}$, s and ϕ in ParRB±ATL*, we have explained why we can synthetise a formula ψ such that $\mathfrak{M}, s \models \mathfrak{v}(\phi)$ iff $\mathfrak{v} \models \psi$ for all interpretations $\mathfrak{v}$ for the resource parameters. Moreover ψ is a Boolean combination of atomic constraints of the form either $\mathtt{x} \geq k$ or $\mathtt{x} = \omega$.
3. The model-checking problem for RBTL* introduced in [BF09] is EXPSPACE-complete. No complexity upper bound for RBTL was known so far as well as the decidability status for RBTL*.

We have been also able to provide complexity results for fragments and for variants, but we believe that the simple framework we have proposed could be used to obtain further results for new resource-bounded logics, which is clearly part of future work.

References

ABLN15. N. Alechina, N. Bulling, B. Logan, and H.N. Nguyen. On the boundary of (un)decidability: Decidable model-checking for a fragment of resource agent logic. In *IJCAI'15*, pages 1494–1501. AAAI Press, 2015.

AHK02. R. Alur, Th. Henzinger, and O. Kupferman. Alternating-time temporal logic. *Journal of the Association for Computing Machinery*, 49(5):672–713, 2002.

ALNR14. N. Alechina, B. Logan, H.N. Nguyen, and F. Raimondi. Decidable model-checking for a resource logic with production of resources. In *ECAI'14*, volume 263 of *Frontiers in Artificial Intelligence and Applications*, pages 9–14. IOS Press, 2014.

ALNR15. N. Alechina, B. Logan, H.N. Nguyen, and F. Raimondi. Technical report: Model-checking for resource-bounded ATL with production and consumption of resources. *CoRR*, abs/1504.06766, 2015.

AMSS13. P.A. Abdulla, R. Mayr, A. Sangnier, and J. Sproston. Solving Parity Games on Integer Vectors. In *CONCUR'13*, volume 8052 of *LNCS*, pages 106–120. Springer, 2013.

BBH+06. A. Bouajjani, M. Bozga, P. Habermehl, R. Iosif, and P. Moro nd T. Vojnar. Programs with lists are counter automata. In *CAV'06*, volume 4144 of *LNCS*, pages 517–531. Springer, 2006.

BF09. N. Bulling and B. Farwer. Expressing properties of resource-bounded systems: The logics RBTL* and RBTL. In *CLIMA X*, volume 6214 of *LNCS*, pages 22–45. Springer, 2009.

BFG+15. M. Blondin, A. Finkel, S. Göller, C. Haase, and P. McKenzie. Reachability in two-dimensional vector addition systems with states is PSPACE-complete. In *LICS'15*, pages 32–43. ACM Press, 2015.

BHSS12. B. Bérard, S. Haddad, M. Sassolas, and N. Sznajder. Concurrent games on VASS with inhibition. In *CONCUR'12*, volume 7454 of *LNCS*, pages 39–52. Springer, 2012.

BJK10. T. Brázdil, P. Jancar, and A. Kucera. Reachability games on extended vector addition systems with states. In *ICALP'10*, volume 6199 of *LNCS*, pages 478–489. Springer, 2010.

BS11. M. Blockelet and S. Schmitz. Model-checking coverability graphs of vector addition systems. In *MFCS'11*, volume 6907 of *LNCS*, pages 108–119. Springer, 2011.

CS14. J.B. Courtois and S. Schmitz. Alternating vector addition systems with states. In *MFCS'14*, volume 8634 of *LNCS*, pages 220–231. Springer, 2014.

CSW16. T. Chen, F. Song, and Z. Wu. Global model checking on pushdown multi-agent systems. In *AAAI'16*, pages 2459–2465. AAAI Press, 2016.

DDMM12. Ph. Darondeau, S. Demri, R. Meyer, and Ch. Morvan. Petri net reachability graphs: Decidability status of FO properties. *Logical Methods in Computer Science*, 8(4), 2012.

Dem13. S. Demri. On selective unboundedness of VASS. *Journal of Computer and System Sciences*, 79(5):689–713, 2013.

Dic13. L.E. Dickson. Finiteness of the odd perfect and primitive abundant numbers with n distinct prime factors. *Amer. Journal Math.*, pages 413–422, 1913.

DJLL13. S. Demri, M. Jurdziński, O. Lachish, and R. Lazić. The covering and boundedness problems for branching vector addition systems. *Journal of Computer and System Sciences*, 79(1):23–38, 2013.

Eme90. A. Emerson. Temporal and modal logic. In *Handbook of Theoretical Computer Science*, pages 996–1072. Elsevier, 1990.

Esp94. J. Esparza. On the decidability of model checking for several μ-calculi and Petri nets. In *ICALP'94*, volume 787 of *LNCS*, pages 115–129. Springer, 1994.

Esp98. J. Esparza. Decidability and complexity of Petri net problems — an introduction. In *Advances in Petri Nets 1998*, volume 1491 of *LNCS*, pages 374–428. Springer, 1998.

GL13. S. Göller and M. Lohrey. Branching-time model checking of one-counter processes and timed automata. *SIAM Journal of Computing*, 42(3):884–923, 2013.

Haa12. C. Haase. *On the Complexity of Model Checking Counter Automata*. PhD thesis, University of Oxford, 2012.

Hab97. P. Habermehl. On the complexity of the linear-time mu-calculus for Petri nets. volume 1248 of *LNCS*, pages 102–116. Springer, 1997.

HR89. R.R. Howell and L.E. Rosier. Problems concerning fairness and temporal logic for conflict-free petri nets. *Theoretical Computer Science*, 64:305–329, 1989.

Jan90. P. Jančar. Decidability of a temporal logic problem for petri nets. *Theoretical Computer Science*, 74(1):71–93, 1990.

JLS15. M. Jurdzinski, R. Lazić, and S. Schmitz. Fixed-dimensional energy games are in pseudo-polynomial time. In *ICALP'15*, volume 9135 of *LNCS*, pages 260–272. Springer, 2015.

KM69. R.M. Karp and R.E. Miller. Parallel program schemata. *Journal of Computer and System Sciences*, 3(2):147–195, 1969.

Lip76. R.J. Lipton. The reachability problem requires exponential space. Technical Report 62, Department of Computer Science, Yale University, 1976.

LMO08. F. Laroussinie, N. Markey, and G. Oreiby. On the expressiveness and complexity of ATL. *Logical Methods in Computer Science*, 4(2), 2008.

LS15. J. Leroux and S. Schmitz. Demystifying reachability in vector addition systems. In *LICS'15*, pages 56–67. IEEE, 2015.

MNP11. D. Della Monica, M. Napoli, and M. Parente. On a logic for coalitional games with priced-resource agents. *ENTCS*, 278:215–228, 2011.

Rac78. C. Rackoff. The covering and boundedness problems for vector addition systems. *Theoretical Computer Science*, 6(2):223–231, 1978.

RSB05. J.F. Raskin, M. Samuelides, and L. Van Begin. Games for counting abstractions. *ENTCS*, 128(6):69–85, 2005.

Sch03. Ph. Schnoebelen. The complexity of temporal logic model checking. In *AIML'02*, pages 437–459. King's College Publication, 2003.

Sch09. S. Schewe. Tighter bounds for the determinisation of Büchi automata. In *FOSSACS'09*, volume 5504 of *LNCS*, pages 167–181. Springer, 2009.

Ser06. O. Serre. Parity games played on transition graphs of one-counter processes. In *FOSSACS'06*, volume 3921 of *LNCS*, pages 337–351. Springer, 2006.

Ves15. S. Vester. On the complexity of model-checking branching and alternating-time temporal logics in one-counter systems. In *ATVA'15*, volume 9364 of *LNCS*, pages 361–377. Springer, 2015.

VGL05. K.N. Verma and J. Goubault-Larrecq. Karp-Miller Trees for a Branching Extension of VASS. *Discrete Mathematics and Theoretical Computer Science*, 7:217–230, 2005.

VW94. M. Vardi and P. Wolper. Reasoning about infinite computations. *Information and Computation*, 115:1–37, 1994.